VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SOUBOR PROGRAMŮ PRO PODPORU VÝUKY ZPRACOVÁNÍ SIGNÁLŮ A OBRAZŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

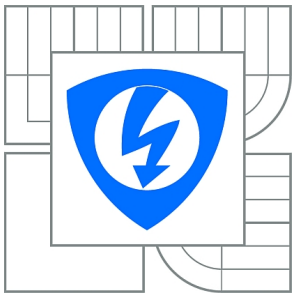
AUTOR PRÁCE
AUTHOR

Bc. MIROSLAV RIŠKO

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

SOUBOR PROGRAMŮ PRO PODPORU VÝUKY ZPRACOVÁNÍ SIGNÁLŮ A OBRAZŮ

BUNDLE OF PROGRAMS FOR TEACHING SIGNAL AND IMAGE PROCESSING

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

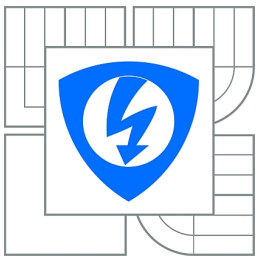
AUTOR PRÁCE — Bc. MIROSLAV RIŠKO

AUTHOR

VEDOUCÍ PRÁCE — Mgr. PAVEL RAJMIC, Ph.D.

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Miroslav Riško ***ID:*** 115267
Ročník: 2 ***Akademický rok:*** 2013/2014

NÁZEV TÉMATU:

Soubor programů pro podporu výuky zpracování signálů a obrazů

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit programy pro interaktivní podporu výuky zejm. v kurzech BZSG, MGMP, BASS. Budou mít podobu JAVA apletů, které budou tématicky zaměřeny na: 1/lineární kombinaci signálů, 2/ aritmetické kódování, 3/ gama korekci, 4/ použití palety pro reprezentaci bitmapového obrazu. Student aplety navrhne, implementuje a ověří jejich funkčnost.

DOPORUČENÁ LITERATURA:

[1] Beneš, B.; Sochor, J.; Felkel, P.; Žára, J.: Moderní počítačová grafika. Computer Press, Brno, 2005.

[2] R. C. Gonzales, R. E. Woods, Digital Image Processing, Third Edition, Prentice Hall, 2008

[3] Schildt, H. : Java7, výukový kurz, Computer Press, Brno, 2012.

Termín zadání: 10.2.2014 ***Termín odevzdání:*** 28.5.2014

Vedoucí práce: Mgr. Pavel Rajmic, Ph.D.
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Cieľom mojej diplomovej práce je vytvorenie programov, ktoré budú slúžiť ako podpora vo výučbe. Práca je rozdelená na dve časti - teoretickú a praktickú. V teoretickej časti sa budem zaoberať teóriou potrebnou k vypracovaniu tejto práce. V praktickej časti bude opísaná samotná funkcionalita vytvorených appletov.

KĽÚČOVÉ SLOVÁ

Signál, applet, java, gama, aritmetické kódovanie

ABSTRACT

The aim of my diploma project is to create a program that will serve as a support for the teaching. Project is divided into two parts - theoretical and practical. In the theoretical part I will discuss the theory necessary for development of this work. The practical part will describe functionality of created applets.

KEYWORDS

Signal, applet, java, gamma, arithmetic coding

RIŠKO, Miroslav *Soubor programů pro podporu výuky zpracování signálů a obrazů*: diplomová práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 61 s. Vedúci práce bol Mgr. Pavel Rajmic, Ph.D.

PREHLÁSENIE

Prehlasujem, že som svoju diplomovoú prácu na tému „Soubor programů pro podporu výuky zpracování signálů a obrazů“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia § 152 trestného zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Technicka 12, CZ-61600 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

ZOZNAM OBRÁZKOV

1 ÚVOD

V dnešnej dobe sa čoraz častejšie vo výučbe začína využívať aktívnejšie zapojenie študenta do výučby rôznymi interaktívnymi pomôckami. Vďaka týmto pomôckam vytvára učiteľ priestor na aktívnu prácu žiaka. Základnou myšlienkou je, aby si žiak sám vytvoril obraz o celku na základe vlastných prežitých skúsenosti pri riešení daného problému a informácií poskytnutými učiteľom.

V porovnaní s minulosťou má dnes učiteľ v rukách silný nástroj, ktorý mu môže prácu zefektívniť alebo ju posunúť na novú kvalitatívnu úroveň. Sú to informačno-komunikačné technológie a na nich založené moderné vyučovacie prostriedky.[2]

Táto diplomová práca sa bude zaoberať vytvorením webových JAVA appletov, ktoré majú pomôcť pri výučbe predmetov zameraných na spracovanie signálu a obrazu. Budú vytvorené celkovo 4 applety.

Zameranie appletov bude nasledovné. V prvom applete sa budem zaoberať kombináciou rôznych signálov, kde si užívateľ bude môcť zvoliť z dvoch typov kombinácií a šiestich rôznych signálov, pri ktorých môže nastavovať ich váhu. V druhom applete sa budem zaoberať úpravou obrazu pomocou gama korekcie. Užívateľ si bude môcť vyskúšať ako sa zmení zvolený obrázok pri zmene hodnoty nastavenej gamy. V predposlednom applete si môže užívateľ vyskúšať aritmetické kódovanie a dekódovanie, kde môže zadať vstupné slovo a sledovať ako sa po kroku zmení na binárne výstupné slovo a následne dekóduje. V poslednom applete bude môcť užívateľ vykresliť maticu rôznymi farbami pomocou indexácie farieb.

2 TEORETICKÝ ÚVOD

V tejto časti práce sa oboznámime s teóriou potrebnou na pochopenie tejto diplomovej práce. Oboznámime sa s:

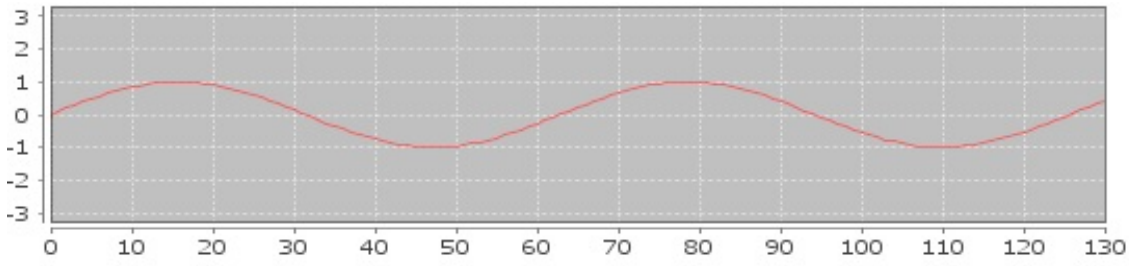
- použitými signálmi v práci
- typmi kombinácií
- barycentrickými súradnicami
- tým ako funguje gama korekcia
- s princípom aritmetického kódovania
- s použitým softwarom v tejto práci

2.1 Typy signálov

V tejto časti práce sa budem zaoberať diskrétnymi signálmi (definovanými v diskrétnych časových okamihoch, tvorených postupnosťou funkčných hodnôt), s ktorými ďalej pracujem v tejto práci.

2.1.1 Sínus

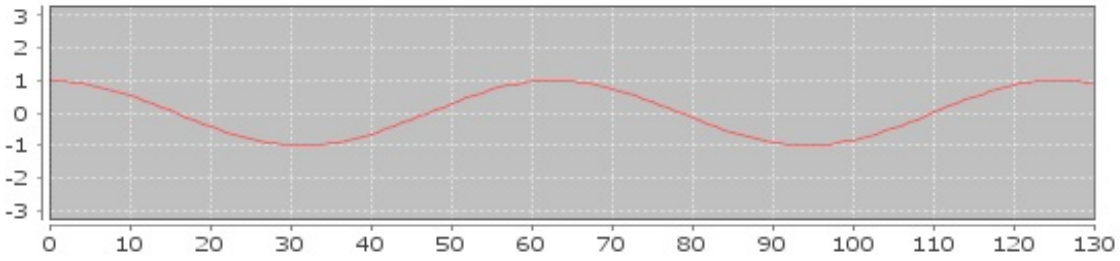
Sínus (viď obr. 2.1) patrí medzi goniometrické funkcie. V pravouhlom trojuholníku je definovaný ako pomer dĺžky protiľahlej odvesny k uhlu a dĺžky prepony trojuholníka. Graf funkcie sínus sa nazýva sínusoida alebo sínusovka. V našom prípade zobrazujeme graf v 2 periódach.



Obr. 2.1: Graf signálu sínus

2.1.2 Kosínus

Kosínus (viď obr. 2.2) patrí medzi goniometrické funkcie. V pravouhlom trojuholníku je definovaný ako pomer dĺžky priľahlej odvesny a dĺžky prepony trojuholníka. Graf funkcie kosínus sa nazýva kosínusoida. Táto funkcia sa zvyčajne označuje skratkou cos. V našom prípade zobrazujem graf v 2 periódach.



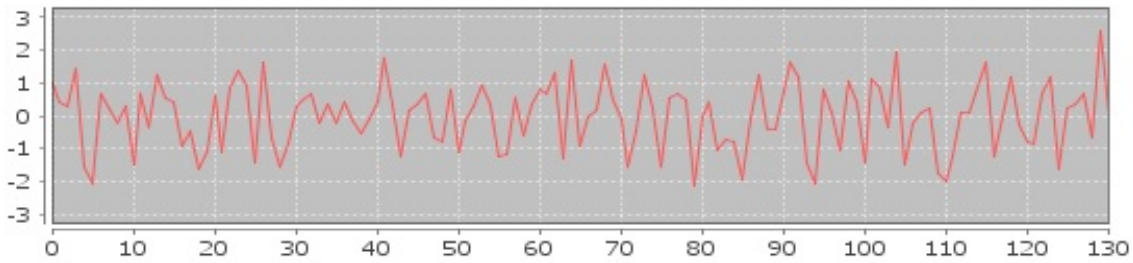
Obr. 2.2: Graf signálu kosínus

2.1.3 Gaussov šum

Gaussov šum (viď obr. 2.3) predstavuje štatistický šum, ktorý má *funkciu hustoty pravdepodobnosti normálneho rozdelenia* (známeho tiež ako *Gaussovo rozdelenie*).

Hodnoty, ktoré šum môže nadobúdať sú z normálneho rozdelenia. Funkcia hustoty pravdepodobnosti P nahodnej veličiny Z z normálneho rozdelenia je tvaru: [4]

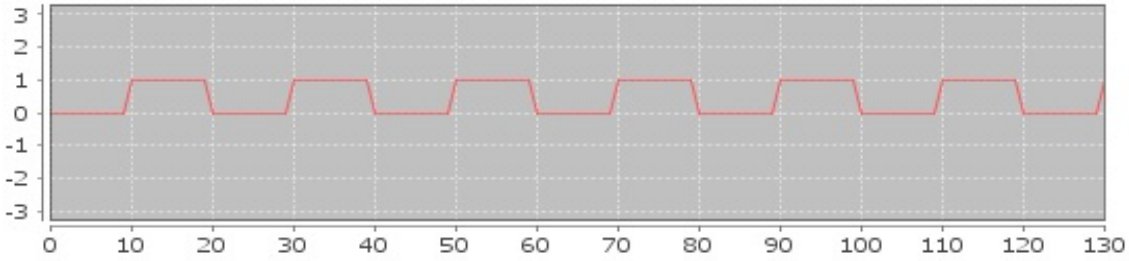
$$P_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(z-\mu)^2}{2\sigma^2}} \tag{2.1}$$



Obr. 2.3: Graf gaussovho šumu

2.1.4 Obdĺžnikový signál

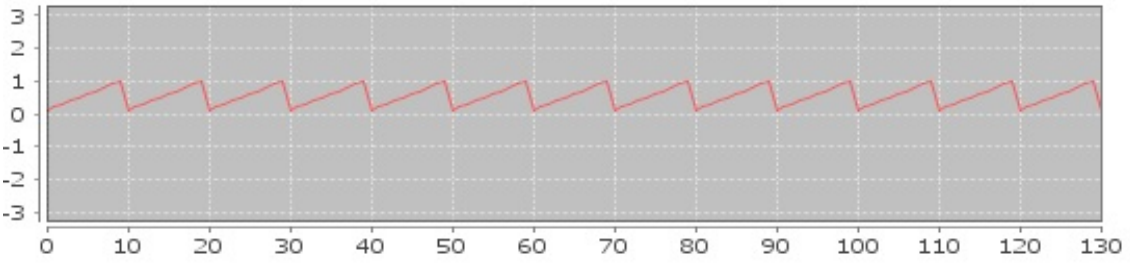
Obdĺžnikový signál (viď obr. 2.4) patrí medzi neharmonické periodické signály. Jeho signál môžeme rozdeliť na dve časti. V prvej polovici periódy je hodnota signálu nulová a v druhej polovici nadobudá zvolenú hodnotu. Táto zmena signálu nie je skoková, ale je postupná.



Obr. 2.4: Graf obdĺžnikového signálu

2.1.5 Píla

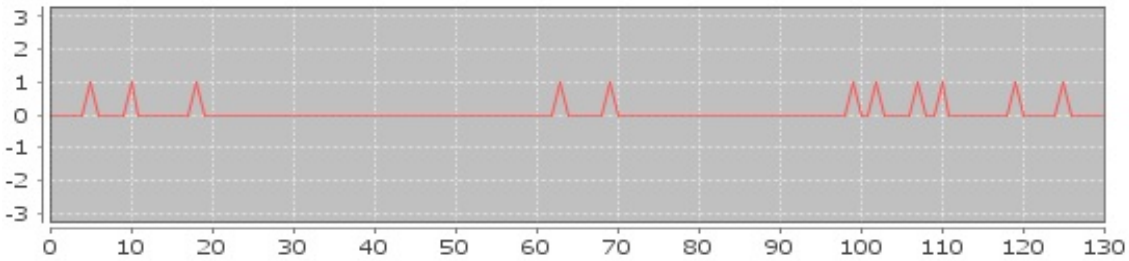
Pílový signál (viď obr. 2.5) patrí medzi neharmonické periodické signály, ktorého veľkosť lineárne stúpa dokým nedosiahne maximálnej hodnoty, aby následné mohla jeho hodnota padnúť na počiatočnú.



Obr. 2.5: Graf pílového signálu

2.1.6 Impulz

Impulzný signál (viď obr. 2.6) je neharmonický neperiodický signál, ktorý je náhodne generovaný v čase t o náhodnom počte n signálov. V tejto práci je možné zvoliť si veľkosť vygenerovaných signálov.



Obr. 2.6: Graf impulzného signálu

2.2 Lineárna kombinácia vektorov

Definícia lineárnej kombinácie n vektorov znie:

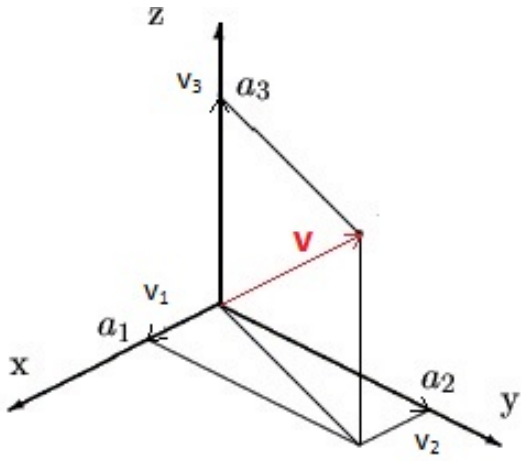
Nech je daných n ľubovoľných vektorov $\vec{v}_1, \vec{v}_1 \ldots, \vec{v}_n$.

Každý vektor v tvaru

$$\vec{v} = a_1\vec{v}_1 + a_2\vec{v}_2 + \ldots + a_n\vec{v}_n \tag{2.2}$$

nazývame *lineárnou kombináciou vektorov* (viď obr. 2.7) $\vec{v}_1, \vec{v}_1 \ldots, \vec{v}_n$, kde reálne čísla $a_1, a_2, \ldots, a_n$ nazývame koeficienty lineárnej kombinácie.

Vieme povedať, že lineárna kombinácia vektorov je jednoduché násobenie a sčítanie, kde vektory násobíme ľubovoľným reálnym číslom a následne tieto násobky sčítame.[17]



Obr. 2.7: Lineárna kombinácia vektorov

2.3 Konvexná kombinácia vektorov

Definícia pre konvexnú kombináciu n vektorov znie:

Nech je daných n ľubovoľných vektorov $\vec{v}_1, \vec{v}_1 \ldots, \vec{v}_n$ a reálne čísla $a_1, a_2, \ldots, a_n \in R$. Potom výsledný vektor

$$\vec{v} = a_1\vec{v}_1 + a_2\vec{v}_2 + \ldots + a_n\vec{v}_n, \tag{2.3}$$

kde $a_1 + a_2 + \ldots + a_n = 1$ a naviac platí:

$$a_i \geq 0 \qquad i = 1, 2, \ldots, n$$

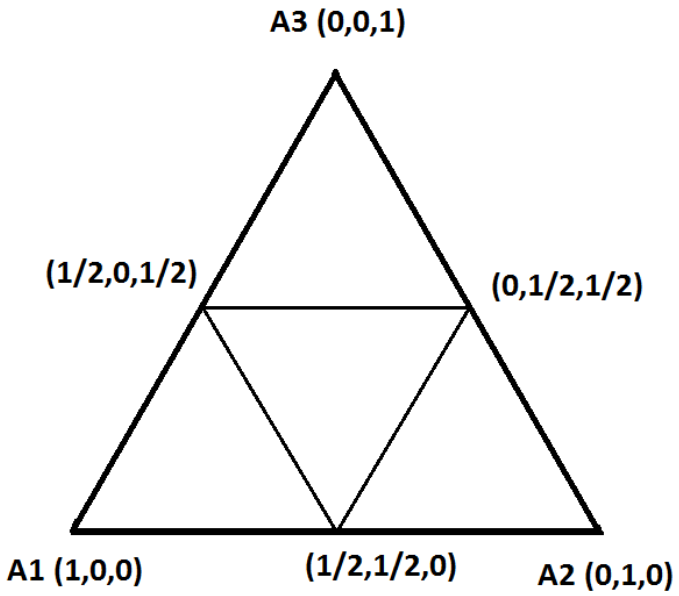
nazývame *konvexnou kombináciou vektorov* $\vec{v}_1, \vec{v}_1 \ldots, \vec{v}_n$ a reálne čísla $a_1, a_2, \ldots, a_n$ nazývame koeficienty konvexnej kombinácie.

Vieme povedať, že konvexná kombinácia vektorov je jednoduché násobenie a sčítanie, kedy výsledný súčet všetkých koeficientov vektorov je rovný 1, ale iba za podmienky, že všetky koeficienty vektorov sú reálne čísla z intervalu [0 ,1].[3]

2.4 Barycentrické súradnice

Barycentrické súradnice boli objavené v roku 1827 Augustom Ferdinantom Möbiusom. Barycentrické súradnice vyjadruje trojica čísel *(t1, t2, t3)* reprezentujúcich umiestnenie vrchola referenčného trojuholníka *A1, A2, A3*.

Pomocou týchto súradníc môžme určiť bod *P*, ktorý je geometrickým ťažiskom troch hodnôt a je identifikovaný súradnicami *(t1, t2, t3)*. Vrcholy trojuholníka sú dané *(1, 0, 0)*, *(0, 1, 0)*, *(0, 0, 1)*(viď obr. 2.8).[7]



Obr. 2.8: Barycentrické súradnice

V 2D systémoch máme trojuholník definovaný 3 bodmi: *p1(x1,y1)*, *p2(x2,y2)*, *p3(x3,y3)* a jedným bodom *P(x,y)*. Barycentrické súradnice dovoľujú vyjadriť nové súradnice pre bod *P* ako lineárnu kombináciu vektorov *p1, p2, p3*.

Presnejšie, súradnice bodu P si vyjadríme pomocou 3 reálnych skalárov *a*, *b*, *c* následujúcim spôsobom:

$$\begin{aligned} x &= ax_1 + bx_2 + cx_3 \\ y &= ay_1 + by_2 + cy_3 \\ 1 &= a + b + c \end{aligned} \tag{2.4}$$

Spôsob výpočtu pre a, b, c je nasledujúci:

$$\begin{aligned} a &= \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)} \\ b &= \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)} \\ c &= 1 - a - b \end{aligned} \tag{2.5}$$

Bod P leží v trojuholníku, len ak súčasne platia všetky následujúce podmienky:[18]

1. $0 \leq a \leq 1$
2. $0 \leq b \leq 1$
3. $0 \leq c \leq 1$
4. $a + b + c = 1$

2.5 Úprava obrazu pomocou gama korekcie

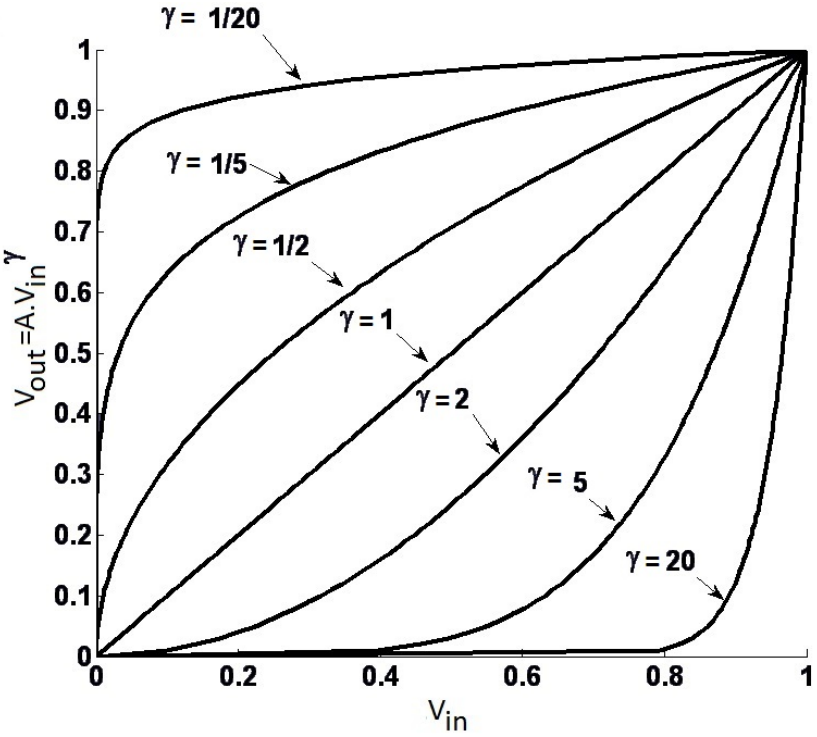
Gama korekcia charakterizuje reprodukciu tónu stupnice v zobrazovaní systému. Gama zahŕňa jednočíselný parameter, nelineárny vzťah medzi hodnotou kódu (od 0 do 255 v 8-bitovom systéme) a jasu. Takmer všetky kódovacie systémy obrazu sú nelineárne, a preto sa hodnoty gamy v jednotlivých systémoch líšia.

Hlavným účelom Gamy korekcie vo videu, či obraze je zmena kódu jasu (v pomere k intenzite) na vnemovo-jednotnú doménu, tak ako optimalizovať vnímanie výkonu obmedzením počtu bitov v každom RGB (alebo CMYK) komponentu.

Vzťah medzi vstupným signálom a výstupným signálom (viď obr. 2.9) je definovaný rovnicou[13]:

$$V_{out} = AV_{in}^{\gamma} \tag{2.6}$$

Kde A je konštanta a vo väčšine prípadov napodúba hodnotu A =1. Hodnoty V_{out} a V_{in} sú typický z rozsahu 0−1 a γ je hodnota samotnej gamy korekcie.



Obr. 2.9: Graf gamy korekcie

2.6 Aritmetické kódovanie

Aritmetické kódovanie je alternatívou k Huffmanovmu kódovaniu. Odstraňuje niektoré jeho nedostatky – oddeľuje pravdepodobnostný model zdroja od procesu kódovania (preto sa dá ľahšie upraviť na adaptívnu verziu) a nevyžaduje celý počet bitov na kódovanie každého znaku.

Spoločnou črtou aritmetického a Huffmanovho kódovania je rovnaký model zdroja – pravdepodobnosti výskytov znakov zdrojovej abecedy. Keďže sa pri kódovaní nevyužívajú žiadne kontextové informácie (pozičné závislosti znakov), zvyknú sa označovať ako „zero-order coders“. Do tejto skupiny patrí aj Shannonov-Fanov kód.[15]

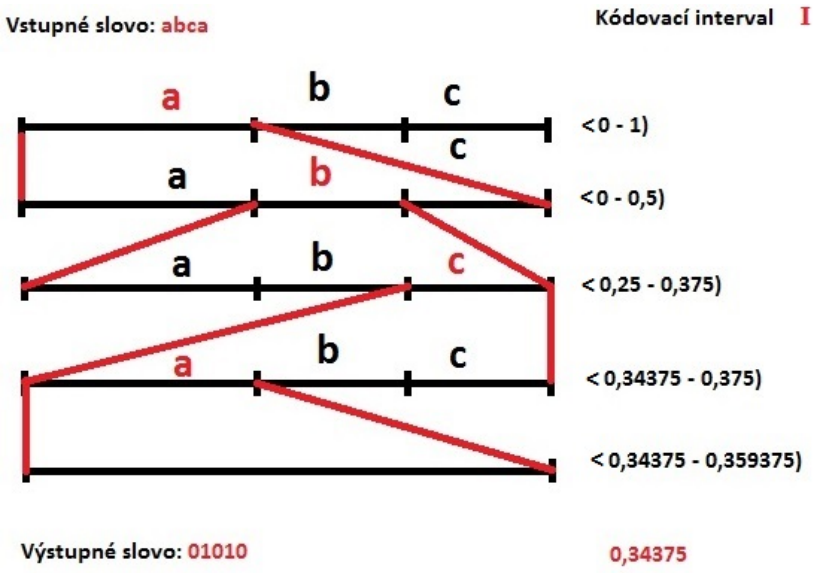
Aritmetické kódovanie je špeciálne kódovanie, ktoré nekóduje jednotlivé symboly, ale vytvára kódové slovo pre celú správu. Vstupom do aritmetického kódovania je vstupný text z abecedy zdrojových jednotiek, výstupom je reálne číslo z intervalu 0 (vrátane) až 1.

Pretože všetky čísla zo zadaného intervalu začínajú 0, nie je nutné túto nulu vkladať do kódového slova. U aritmetického kódovania je nutné ďalej uložiť informáciu o počte zakódovaných čísel, aby dekódovací algoritmus včas skončil s de-

lením intervalu. To je možné vyriešiť napríklad dodaním špeciálneho symbolu označujúceho koniec vstupu alebo skorším explicitným uvedením počtu zakódovaných symbolov.[5]

2.6.1 Pseudokód

1. Do výstupu ulož pravdepodobnosť alebo početnosť symbolov.
2. Nastav interval I = <0,1).
3. Pokiaľ nie je zakódovaný celý text, opakuj:
   - Načítaj ďalší symbol C daného vstupného slova.
   - Rozdeľ interval I na podintervaly, ich veľkosti sú úmerné pravdepodobnosti symbolov.
   - Do I ulož podinterval odpovedajúci symbolu C.
4. Výstupom je ľubovoľné číslo z intervalu I bez počiatočnej nuly [5](viď obr. 2.10).



Obr. 2.10: Graf gamy korekcie

2.6.2 Výstupné slovo

Výstupné slovo aritmetického kódovania sa zvolí z výsledneho intervalu (viď obr. 2.10). Následne sa prevádza do binárneho tvaru, ktorý však môže obsahovať veľké množstvo znakov. Preto sa výsledné slovo orezáva a to podľa nasledujucého vzťahu:

$$\log_2 \left\lceil \frac{1}{koniec\_interval - start\_interval} \right\rceil + 1 = X, \tag{2.7}$$

kde X označuje dekadické číslo určujúce na koľko znakov sa oreže binárne číslo(číslo sa zaokruhľuje smerom dole).

2.6.3 Dekódovanie

Pri dekódovaní nie je potrebné predávať výsledný interval, stačí predávať len zvolené slovo, prípadne číslo (ďalej v texte označené ako n). Musíme mať, ale k dispozícii pravdepodobnosť výskytu znakov abecedy.

Pri dekodovaní postupujem rovnako analogicky ako pri kódovaní. Na začiatku nastavím interval $I =< 0, 1 >$ (spodná hranica, horná hranica). V tomto intervale sa nachádza čislo n. Pre zistenie prvého znaku je potrebné určiť, v ktorom z potencionálnych intervalov $I_1, \ldots, I_k$ sa číslo n nachádza. Následne toto číslo n upravujem, kedy pri každom kroku k nemu nájdem príslušný znak podľa intervalu, v ktorom práve „leží“ n. Hodnota n sa vypočítava z nasledujúceho vzťahu:

$$n = \frac{n - spodna\_hranica}{horna\_hranica - spodna\_hranica} \tag{2.8}$$

Celý cyklus opakujem tak dlho, dokým nedekodujem celý reťazec znakov[12].

3 POPIS POUŽITÉHO SOFTWARU

3.1 Java

Java je programovací jazyk vyvinutý firmou *Sun Microsystem*, neskôr kúpený firmou *Oracle*. Je to objektovo orientovaný jazyk vychádzajúci z C++, ku ktorému má taktiež syntakticky najbližšie, ak teda nepočítame C#, ktorý vznikol až po príchode Javy.

Oproti svojmu predchodcovi, Java neobsahuje niektoré konštrukcie, ktoré spôsobovali pri programovaní v C++ najväčšie problémy a navyše pridáva mnoho užitočných vlastností, napr.:

- Prideľovanie a uvoľňovanie pamätí je tu obstarané automaticky (pomocou garbage collectoru). Objekt sa neruší pomocou *delete* alebo *free ()*, iba sa „ponúkne“ ku zrušeniu (napr. priradením neplatnej referencii null).
- Klasický problém ukazovateľa z C/C++, je tu celkom odstránený, pretože ten sa tu jednoducho nenachádza (resp. nahradený referenciami). Dereferencovanie má samozrejme na starosť runtime. Programátor je tak ušetrený notorickej chyby zápisu pointeru mimo dátovú oblasť.
- Je implementovaný mechanizmus vlákien (*threads*) a je teda možné spúšťať viac úloh v rámci jedného programu. Nezabudlo sa samozrejme ani na ich synchronizáciu pomocou tzv. *monitorov*.
- Vytvorené objekty sa dajú automaticky serializovať, tj. ukladať do súboru, zasielať po sieti a pod.
- Je možné vykonávať reflexiu, čiže zisťovanie informácií o objekte (aké ma premenné, metódy a pod.).
- Implementovaný mechanizmus výnimiek, takže všetky runtime chyby je možné odchytiť a spracovať. Výnimky sú samozrejme objektové, takže je možné zachytiť aj celú hierarchiu výnimiek v jednom pozorovanom bloku.
- Značným uľahčením práce pre programátorov je obsiahlosť štandardne dodávaných knižníc, s ktorými sa nemôže zrovnávať pravdepodobne žiaden bežne používaný programovací jazyk.
- Java podporuje tvorbu dynamicky rozšíriteľných aplikácií (plug-in a pod.).
- Java kladie značný dôraz na bezpečnosť najmä vďaka tomu, že je prekladaná do bytecodu a implementovaná bezpečnostným mechanizmom (podpisovanie kódov a prideľovanie práv pre rôzne akcie). Je možné zaistiť, že program v Jave, ktorý si užívateľ stiahne zo siete, mu nezformatuje disk, nebude komunikovať so žiadnym iným počítačom, ale iba s počítačom, z ktorého pochádza atď.

- Všetky tieto výhody je možné získať veľmi rýchlo. Java je jazyk veľmi jednoduchý, prehľadný a zrozumiteľný. Pri osvojovaní základov Javy, spraví začiatočník podstatne menej chýb ako pri štúdiu C/C++.
- Veľkou výhodou Javy je taktiež jej hardwarová nezávislosť, pretože je prekladaná do špeciálneho medzikódu (bytecod). Ten je na konkrétnom počítači alebo zariadení (PC, handheld, mobilný telefón a pod.) interpretovaný, prípadne za behu prekladaný do natívneho kódu (tzv. JIT - Just-In-Time compilerom).
- Programátor môže napísať javovský program, napr. na PC pod Windowsom a spustiť ho aj na PC s Linuxom, na Macu, proste všade, kde je k dispozícii Java runtime.

  K dispozícii sú napr. tieto knižnice pre tvorbu:

  1. grafického užívateľského rozhrania (GUI)
  2. vstup/výstup
  3. prácu s textom
  4. komunikáciu s SQL
  5. prácu s komprimovanými súbormi
  6. a iné

  Pre písanie programov je k dispozícii celá rada vývojových prostredí. Najčastejšie sú v praxi používanie tieto:[14]

  1. Eclipse - opensource - zdarma
  2. IDEA - základná verzia zdarma
  3. NetBeans - opensource, zdarma

3.2 Java Applet

Java applet je applet napísaný v programovacom jazyku Java. Sťahuje sa zo servera vo forme Bytecodu. Java applety sa spúšťajú buď vo webovom prehliadači alebo v aplikácií AppletViever od spoločnosti Sun Microsystem.

Pri návšteve webovej stránky, ktorá obsahuje applet, sa applet natiahne do internetového prehliadača. Následne je applet spúšťaný v programe Virtual Java Machine, ktorý sa do počítača inštaluje spolu s prehliadačom.

U niektorých prehliadačov (napr. Microsoft Internet Explorer) nie je inštalácia Virtual Java Machine (terminológia Microsoftu uvádza Microsoft Virtual Machine) obsiahnutá v typickej inštalácii.

Applet môže mať sprístupnené niektoré svoje vlastnosti a metódy ako verejné, čo umožňuje externým programátorom manipuláciu s nimi. [10]

Z bezpečnostných dôvodov platia pre applet niektoré obmedzenia a naopak má applet niektoré funkcie rozšírené:

- Applet nemôže nahrávať knižnice ani definovať natívne metódy.
- Applet nemôže nadväzovať sieťové spojenia na iný ako domovský server.
- Applet nemôže zapisovať do súboru na strane klienta (prehliadača).
- Applet nemôže spúšťať programy na domovskom servery.
- Applet nemôže čítať niektoré systémové premenné.
  (napr. metóda System.getProperty())
- Applet môže prehrávať zvuky.
- Applet môže požiadať browser o zobrazenie ľubovoľnej WWW stránky.
- Applet môže volať verejné metódy appletov umiestnených na tej istej WWW stránke.

Niektoré prehliadače umožňujú uvedené obmedzenia nastaviť individuálne pre vybrané applety.[8]

3.3 Využitie Java appletov

Ako nástroj môžu byť Java applety v rámci daného učebného materiálu použité efektívne, ale aj neefektívne. Efektívnosť ich použitia závisí od mnohých faktorov, napríklad od toho, ako sú zamerané na známe ťažkosti študentov, od toho, ako použijú študenti appletom zobrazené vizuálne informácie, od toho, či je vizualizácia pre daný učebný materiál dôležitá, ale aj od toho, či je applet pre daný materiál vhodný.

Ak sa rozhodneme Java applet použiť ako súčasť úlohy, mal byť daný Java applet pre vyriešenie danej úlohy nevyhnutný. To sa dá dosiahnuť napríklad tak, že údaje potrebné k vyriešeniu danej úlohy musí študent zistiť z appletu. Študent ich teda získava priamym pozorovaním, zbieraním alebo pomocou pomocného výpočtu.

Výhodou takéhoto prístupu je, že študenti si úlohu predstavia (vizualizujú) ešte pred jej samotným riešením. Vizualizácia je dôležitým prvým krokom v každej dobrej stratégii riešenia úloh.

Riešenie ľubovolnej úlohy s Java appletom sa v mnohom podobá laboratórnemu cvičeniu s otvoreným koncom, v ktorom je študentovi zadaná úloha a on sám musí prísť na to, čo zmerať a ako úlohu vyriešiť.

Webová stránka s Java appletom, najčastejšie vystupuje ako počítačová simulácia v rámci interaktívneho učebného materiálu. Takýto interaktívny materiál, ale by mal byť:

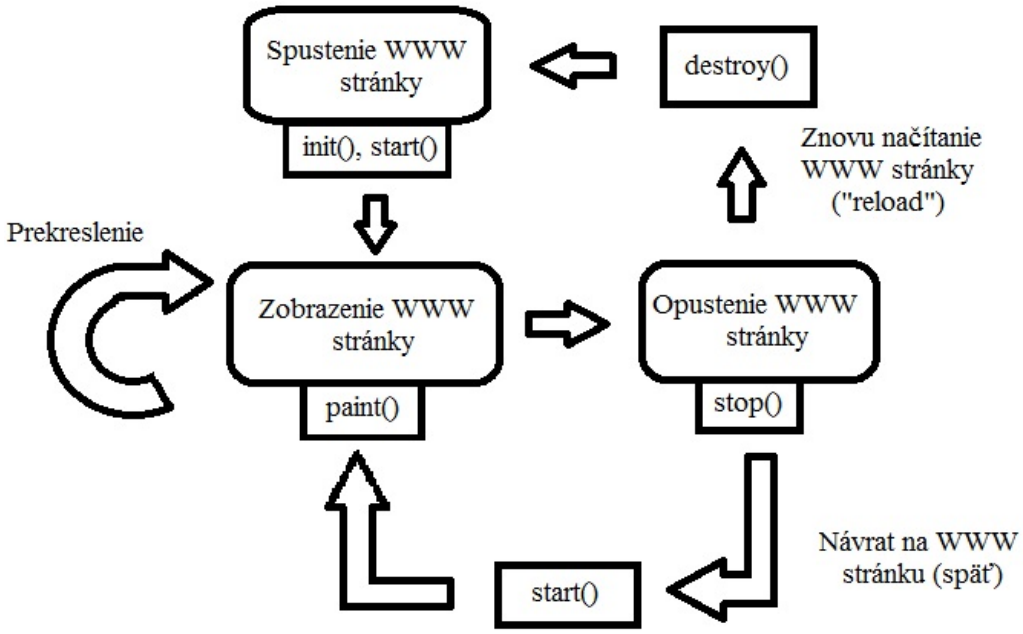
1. Autentický. To znamená, že by mali byť použiteľný v reálnych situáciách. Mal by byť schopný niečo naučiť a to takým spôsobom, aby to študenti, ktorí ho používajú, boli schopní pochopiť.
2. Prevzateľný. To znamená, aby ho učiteľ ľahko mohol začleniť do vyučovania a zároveň študent by nemal mať problém ho využívať, resp. naučiť sa ho ovládať.
3. Prispôsobiteľný. To znamená, že by malo byť ľahké ho modifikovať tak, aby bol využiteľný aj v nových vyučovacích situáciách. Žiadne dve triedy žiakov nie sú rovnaké. Každý učiteľ by mal byť schopný použiť applet v interaktívnom učebnom materiáli podľa svojej chuti, s využitím svojich postupov a metód.

Splnenie prvých dvoch podmienok má na starosti samotný tvorca Java appletov. Posledná podmienka, prispôsobiteľnosť, je v prípade appletov tiež splnená. Učiteľ môže webovú stránku obsahujúcu Java applet, zmeniť ľahko podľa svojho vkusu, jednoducho tak, že zasiahne do html kódu stránky s appletom. Iný spôsob, ako možno ovplyvniť funkciu Java appletu na stránke je využiť skriptovanie, teda spoluprácu Java appletu s Java Scriptom v html súbore[16].

3.4 Štruktúra appletu

Základnú štruktúru appletu tvorí trieda *java.applet.Applet*, ktorá definuje základné metódy tvoriace rozhranie medzi prehliadačom a appletom. Program, ktorý má fungovať ako applet, musí byť potomkom tejto triedy. Applet je spustený v grafickom kontexte a je úzko spojený s „oknovou“ knižnicou AWT.

Trieda *java.applet.Applet* je potomkom triedy *java.awt.Panel*, ktorá umožňuje appletu vlastniť komponenty užívateľského rozhrania, vykonávať grafický výstup a zachytávať udalosti z klávesnice a myši. Životný cyklus appletu (viď obr. 3.1) závisí na prehliadači, ktorý v priebehu svojej činnosti volá tieto metódy appletu:



Obr. 3.1: Životný cyklus appletu

- public void init () - pri inicializácií
- public void start () - pri spustení
- public void paint (java.awt.Graphics g) - pri prekresľovaní
- public void stop () - pri zastavení
- public void destroy () - pri ukončení

Tieto metódy sú v triede *java.applet.Applet* definované ako prázdne a pre zmysluplnú činnosť je potrebné v potomkovi prekryť aspoň jednu z nich. Kostra appletu, ktorý implementuje všetky základne metódy, vyzerá takto:

```
public class NovyApplet extends java.applet.Applet {

public void init() {
// kód prevedený pri inicializácií
}
public void start() {
// kód prevedený pri spustení
```

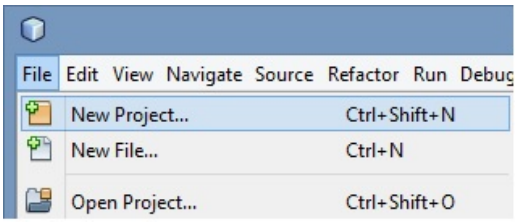
```
}
    public void stop() {
// kód prevedený pri zastavení
}
    public void destroy() {
// kód prevedený pri ukončení
}
    public void paint(java.awt.Graphics g) {
// kód prevedený pri prekresľovaní
    }
}
```

U niektorých prehliadačov môže po zastavení appletu dôjsť automaticky k jeho zrušeniu (po metóde *stop ()* sa bezprostredne volá metóda *destroy ()*).[8]

3.5 Vytvorenie Java appletu

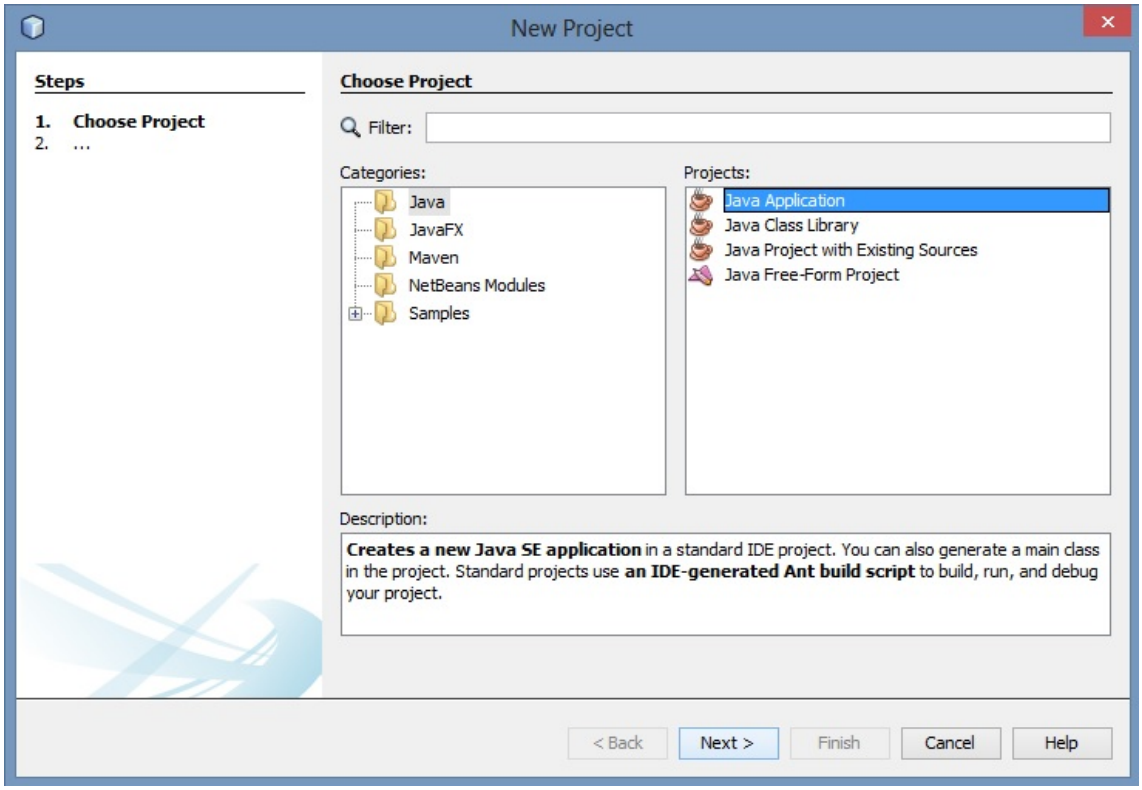
Pri tejto práci som využíval program NetBeans a preto sa tento návod vzťahuje na zmienený program. Prvým krokom pri vytvorení Java appletu je vytvorenie projektu. Po spustení prostredia NetBeans ide sa projekt vytvára následovne:

„File“ → „New Project“ (alebo klávesovou skratkou „Ctrl-Shift-N“) (viď obr. 3.2).

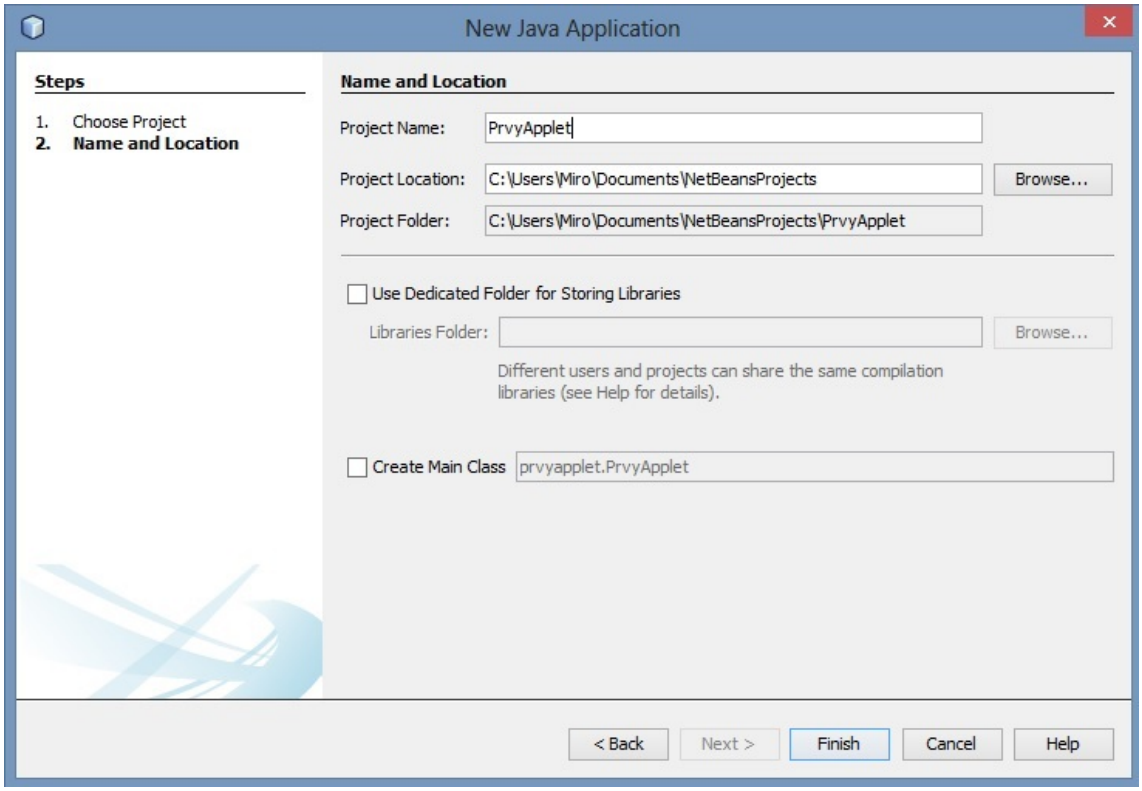


Obr. 3.2: Vytvorenie projektu

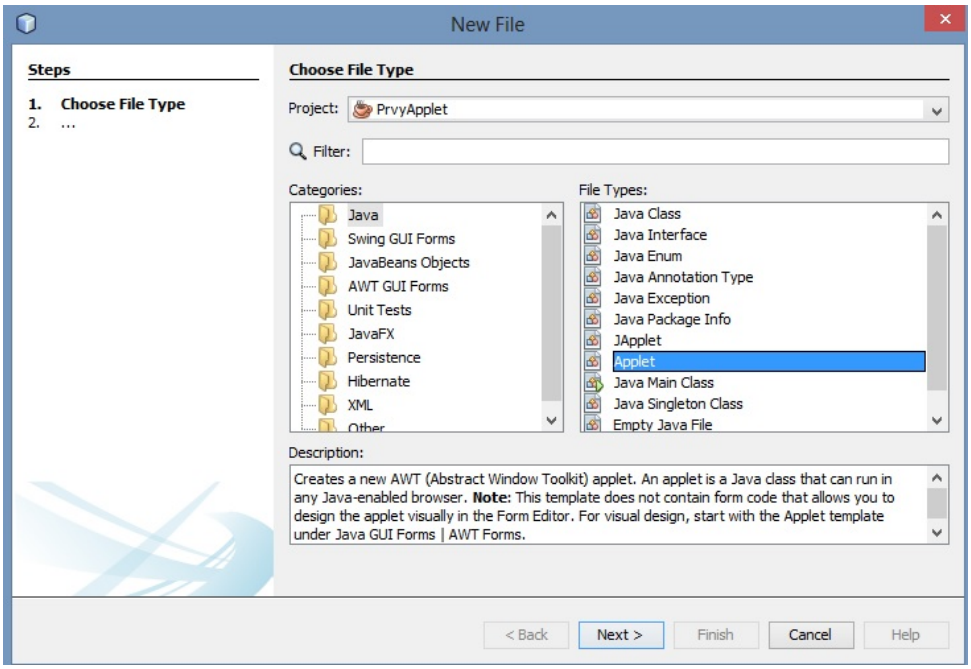
Následne vyskočí „pop-up“ okno tzv. sprievodca vytvorením projektu. V tomto okne zvolíme typ projektu „Java“ a ako typ aplikácie „Java Application“. V spodnej časti tohto okna sa vyskytuje samotný popis danej vytváranej applikacie. Klikneme na tlačidlo „Next >“ (viď obr. 3.3).V druhom kroku si svoj projekt pomenujeme a odškrtneme možnosť „Create Main Class“ a ukončíme vytváranie projektu tlačidlom „Finish“ (viď obr. 3.4).Teraz máme vytvorený projekt a môžeme vytvoriť samotný applet. Do projektu pridáme novú triedu: „File“ → „New File“. Vyberieme kategóriu „Java“ a typ súboru „Applet“(viď obr. 3.5). V ďalšom kroku zvolíme názov appletu (napríklad: „MojPrvyApplet“) a dokončíme sprievodcu.



Obr. 3.3: Typ aplikácie



Obr. 3.4: Dokončenie projektu



Obr. 3.5: Typ súboru

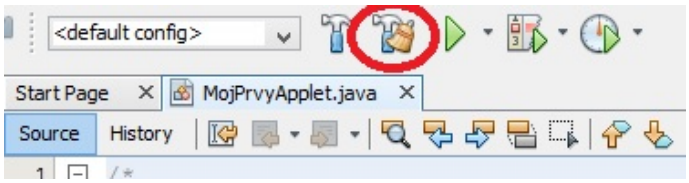
Vytvorí sa nám krátky zdrojový kód. Tento kód si doplníme napríklad metódou „paint“, vďaka ktorej môžeme vykresliť obsah appletu. Zdrojový kód jednoduchého appletu je nasledovný[9].

```
import java.applet.Applet;
import java.awt.*;

public class MojPrvyApplet extends Applet {
   int width, height;
   public void init() {
       width = getSize().width;
       height = getSize().height;
       setBackground(Color.black);
    }
    public void paint(Graphics g) {
       g.setColor(Color.green);
       for (int i = 0; i < 10; ++i) {
          g.drawLine(width, height, i * width / 10, 0);
       }
    }
}
```

Knižnica „import java.awt.*;“ je základná vykreslovacia knižnica, ktorú si Neatbeans doplní sám. V kóde sú deklarované 2 premenné a to: „width a height“. Tieto premenné sú nastavené v metóde „init“. V metóde „paint“ je nastavený cyklus 10 opakovaní, kedy v každom kroku sa určí čiare jej koncová súradnica X pri spoločnej súradnici Y. Farba pozadia je nastavená v metóde „init“ a farba čiar v metóde „paint“.Tento applet si následne môžeme zobraziť pomocou klávesovej skratky „Shift+F6“.

Aby bol applet prístupný pre každého, je potrebné ho umiestniť na FTP server. Pred samotným umiestnením na FTP server je potrebné daný applet skompilovať. V hornej lište (v programe Netbeans) je potrebné kliknúť na „Run“ → „Clean and Build project“ (meno projektu v tomto príklade je PrvyApplet („Shift + F11“)). Ďalšou možnosťou je kliknúť na ikonu znázornenú na obr. 3.6.



Obr. 3.6: Kompilácia projektu

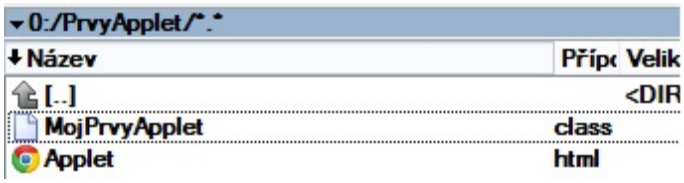
Po tomto kroku sa nám v mieste uloženia projektu vytvoria viaceré súbory. Nás pri nahrávaní appletu na FTP server bude zaujímať nasledovný:

..\PrvyApplet\build\classes\MojPrvyApplet.class

Tento samotný súbor na zobrazenie appletu v internetovom prehliadači nestačí. Je nutné vytvoriť aj jednoduchý HTML súbor, ktorý je následne nutné nahrať na FTP server spolu s predošlým zmieneným súborom. Vytvorený HTML súbor môže vyzerať nasledovne (pomenujeme ho napr. Applet.html):

```
<!DOCTYPE html>
<html>
  <head>
    <title>PrvyApplet< /title>
  < /head>
  <body>
    <applet width=300 height=300 code="MojPrvyApplet.class">
    < /applet>
  < /body>
< /html>
```

Na FTP serveri, by to malo vyzerať, napríklad, ako je znázornené na obrázku 3.7(v mojom prípade som si oba súbory uložil do priečinku, ktorý som pomenoval PrvyApplet).

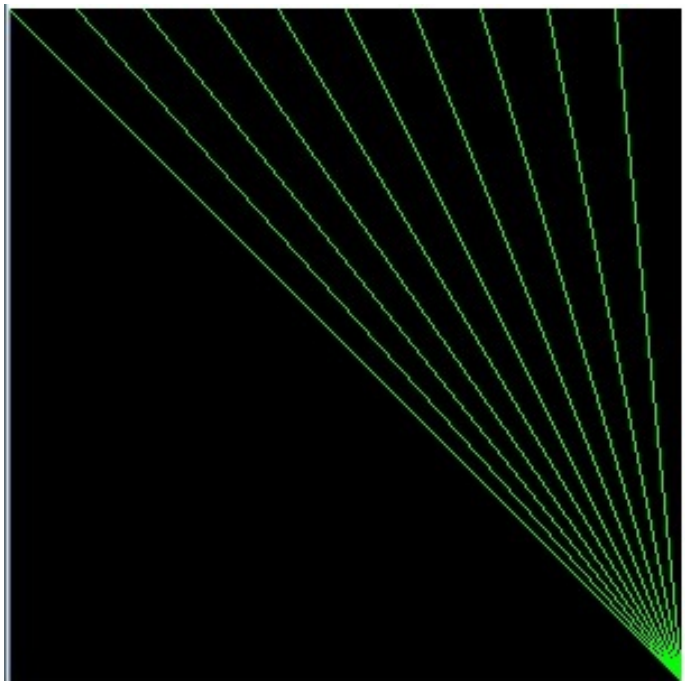


Obr. 3.7: FTP

K samotnému appletu sa potom dostaneme zadaním WWW stránky:

http://adresaFTPservera/PrvyApplet/Applet.html

Samotný applet je následne možné vidieť na obr. 3.8.



Obr. 3.8: PrvyApplet

3.6 Knižnica JFreeChart

Javovská knižnica JFreeChart je zadarmo a slúži na zobrazovanie grafov v profesionálnej kvalite v java aplikáciách. Zahŕňa rozsiahlu sadu funkcií, napr.:

- konzistentné a dobre zdokumentované API, podporujúce širokú škálu typov grafov
- flexibilný dizajn, ktorý je ľahko rozšíriteľný ako na strane servera, tak na strane klienta
- podporuje mnoho typov výstupu, napr.:

    1. swing komponenty
    2. obrazové súbory (PNG,JPEG ...)
    3. formáty vektorovej grafiky (PDS,EPS a SVG)

- JFreeChart je „open source“ alebo lepšie povedané „free software“. Je distribuovaný pod podmienkami GNU Lesser General Public License (LGPL), čo umožňuje použitie vo vlastných aplikáciách.[6]

3.7 NetBeans

NeatBeans je Open Source projekt s veľmi rozsiahlou užívateľskou základňou, rastúcou komunitou vývojárov a takmer 100 partnerov po celom svete. V roku 2000 sa stal hlavný sponzor firma Sun Microsystem.
Dnes existujú dva produkty:

1. Vývojové prostredie NetBeans (NetBeans IDE)
2. Vývojová platforma NetBeans (NetBeans Platform)

Vývojové prostredie NetBeans IDE je nástroj, pomocou ktorého programátori môžu písať, prekladať, ladiť a distribuovať aplikácie. Samotné vývojové prostredie je vytvárané v jazyku Java - avšak podporuje prakticky akýkoľvek programátorský jazyk.

Existuje tak isto veľké množstvo modulov, ktoré toto vývojové prostredie rozširujú. Vývojové prostredie NetBeans je bezplatne šíriteľný produkt a jeho užívanie nie je nijak obmedzené.

Okrem vývojového prostredia je taktiež dostupná vývojová platforma NetBeans Platform, čo je modulárny a rozšíriteľný základ pre vytváranie rozsiahlych počítačových aplikácií. Nezávislí dodávatelia softwaru ponúkajú dodatočné moduly, ktoré je možné ľahko integrovať a taktiež môžu byť použité k vývoju ich vlastných nástrojov a riešení.[11]

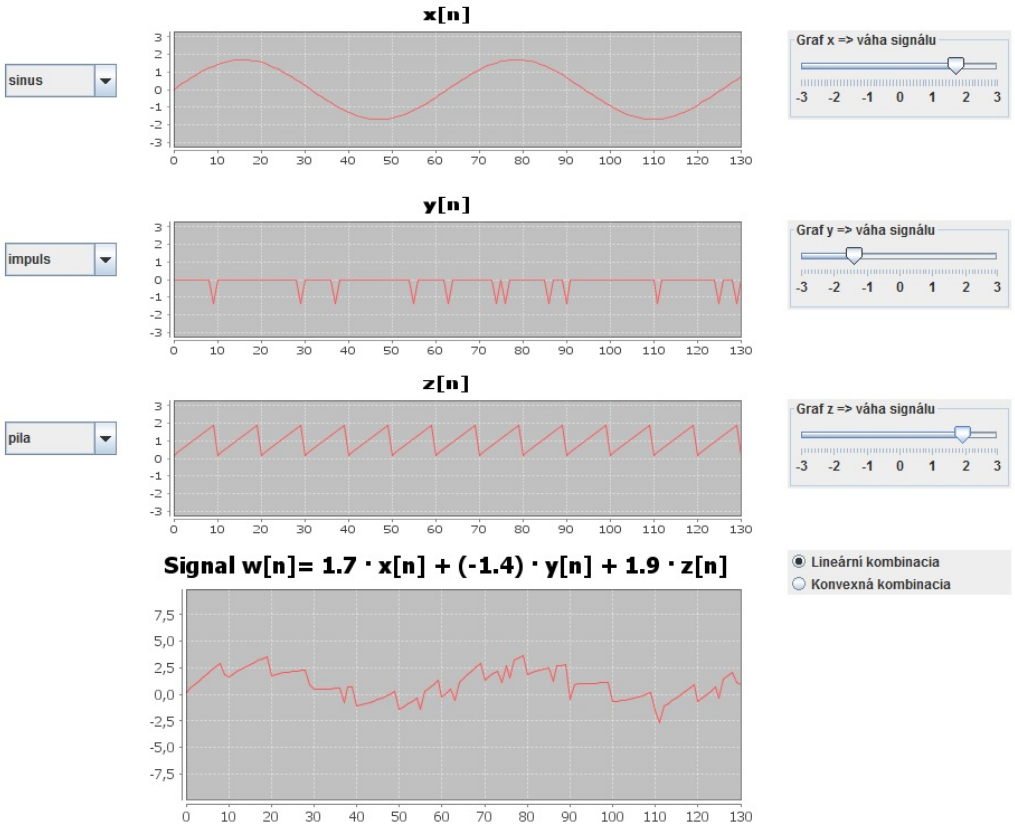
4 PRAKTICKÁ ČASŤ

V praktickej časti mojej diplomovej práci som sa zaoberal vytvorením Java appletov, ktoré majú slúžiť predovšetkým na študijné účely. Táto práca popisuje nasledujúce applety:

1. applet na simuláciu lineárnej a konvexnej kombinácie signálov
2. applet na úpravu obrazu pomocou gama korekcie
3. applet znázorňujúci aritmetické kódovanie a dekódovanie
4. applet s indexáciou farieb

Všetky applety som napísal vo vývojovom prostredí NetBeans. Pri tvorení tejto práce som narazil na niekoľko problémov, ktoré som zaznamenal a ich popis sa nachádza v sekcii s prílohami.

4.1 Applet: Lineárna a konvexná kombinácia



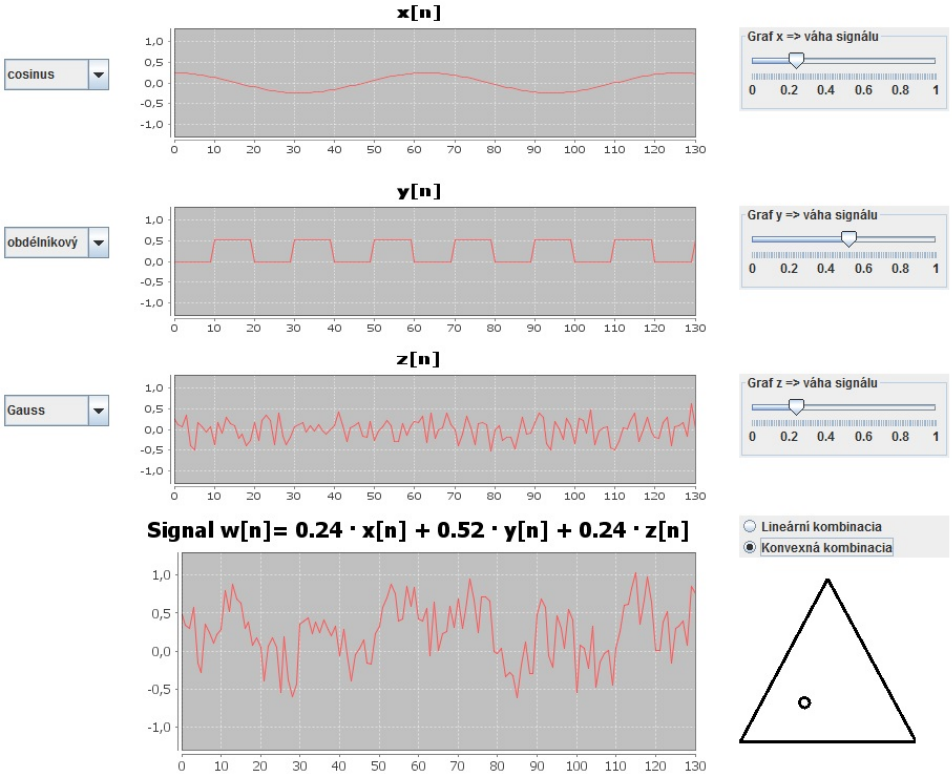
Obr. 4.1: Lineárna kombinácia signálov

V applete je možné simulovať lineárnu kombináciu signálov a konvexnú kombináciu signálov. Užívateľ má možnosť výberu z viacerých druhov signálov, pri ktorých môže nastavovať silu jednotlivých signálov.

4.1.1 Popis appletu

Samotný applet môžeme rozdeliť na 2 hlavné časti:

1. Lineárnu kombináciu (viď obr. 4.1)
2. Konvexnú kombináciu (viď obr. 4.2)



Obr. 4.2: Konvexná kombinácia signálov

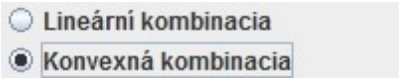
Celý applet by sme si mohli rozdeliť na niekoľko častí:

1. Možnosť voľby typu kombinácie
2. Výber signálu
3. Graf vybraného signálu
4. Posuvník na zmenu veľkosti signálu
5. Barycentrický trojuholník
6. Graf výslednej kombinácie signálov

Možnosť voľby typu kombinácie

Pri spustení appletu je pôvodne nastavené, aby sa signály spočítavali pomocou lineárnej kombinácie.

Pokiaľ chceme prepnúť na konvexnú kombináciu, stačí zaškrtnúť túto voľbu ako je znázornené na obr. 4.3.



Obr. 4.3: Voľba kombinácie

Pri prepínaní medzi jednotlivými kombináciami sa nastavené hodnoty nestrácajú, ale uchovávajú sa v pamäti PC.

Výber signálu

V applete sú 3 okná slúžiace na výber signálu. Pri samotnom štarte appletu majú prednastavené signály. V prvom okne je navolený sínusový signál, v druhom kosínus a v poslednom okne je zvolený signál píla. Každé okno pre výber signálu je pridružené práve k jednému grafu a jednému posuvníku.

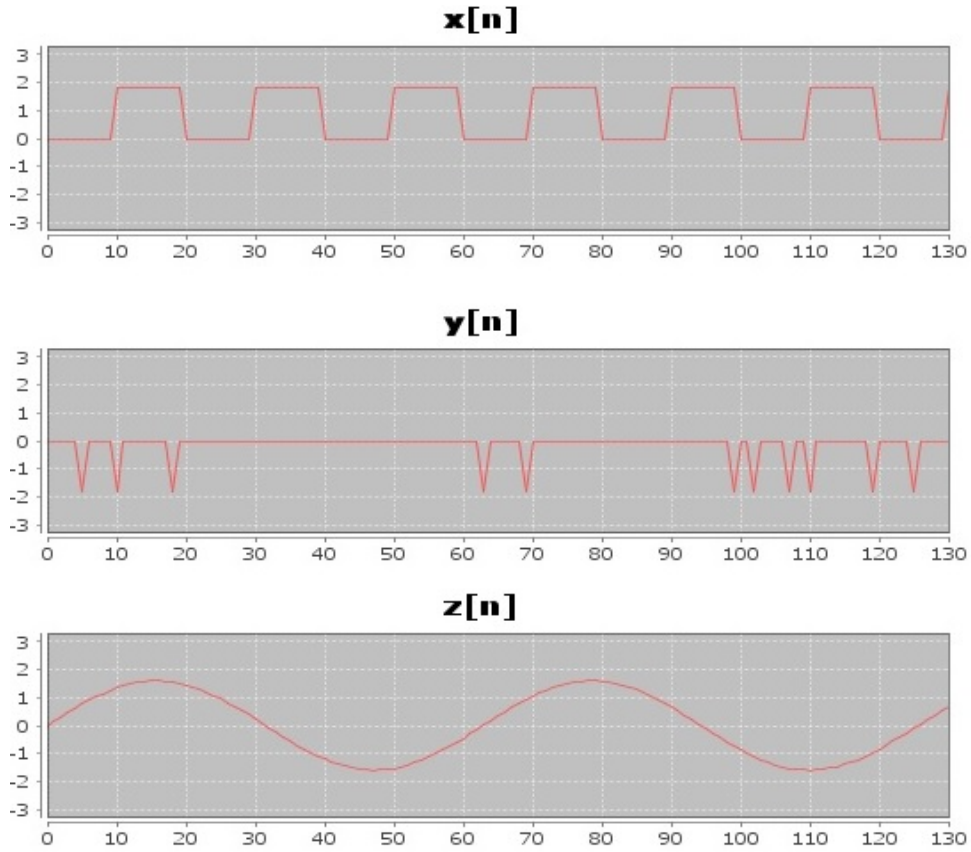
Obr. 4.4: Výber signálu

Celkovo je na výber 6 rôznych signálov (viď obr. 4.4):

1. obdĺžnikový
2. pílový
3. sínosový
4. kosínusový
5. impulsový
6. Gaussov šum

Graf vybraného signálu

Po vybraní daného typu signálu sa ten následne zobrazí v grafe (viď obr. 4.5). Každý graf vykresľuje 131 hodnôt, ktorými je určený daný signál pri rozsahu od -3 do +3. Pre vykresľovanie v Jave bola použitá knižnica *JFreeChart*.

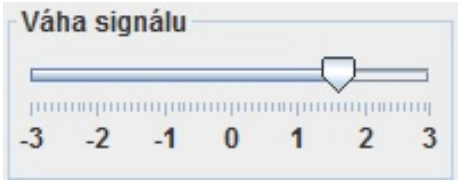


Obr. 4.5: Graf vybraného signálu

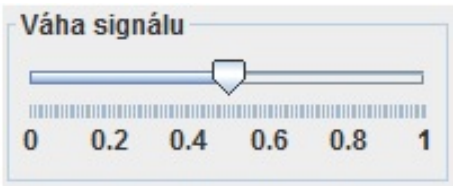
Posuvník na zmenu veľkosti signálu

U každého vybraného signálu je možné meniť jeho váhu. Pri lineárnej kombinácii môžeme upravovať daný signál v rozsahu od -3 do +3 (viď obr. 4.6). Pôvodne je každý posuvník pre všetky signály nastavený na hodnotu 1.

Pri konvexnej kombinácii môžeme upravovať daný signál v rozsahu 0 až 1 (viď obr. 4.7). Prvé dva posuvníky sú prednastavené na hodnotu 0.5 a posledný na hodnotu 0.



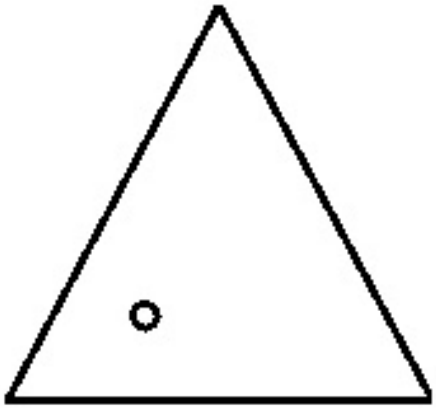
Obr. 4.6: Posuvník pre lineárnu kombináciu



Obr. 4.7: Posuvník pre konvexnú kombináciu

Barycentrický trojuholník

Pri konvexnej kombinácii je aj druhá možnosť ako nastavovať veľkosť váh signálov. Môžeme na to využiť práve barycentrický trojuholník(viď obr. 4.8), kde sa nastavujú váhy pomocou barycentrických súradnic.

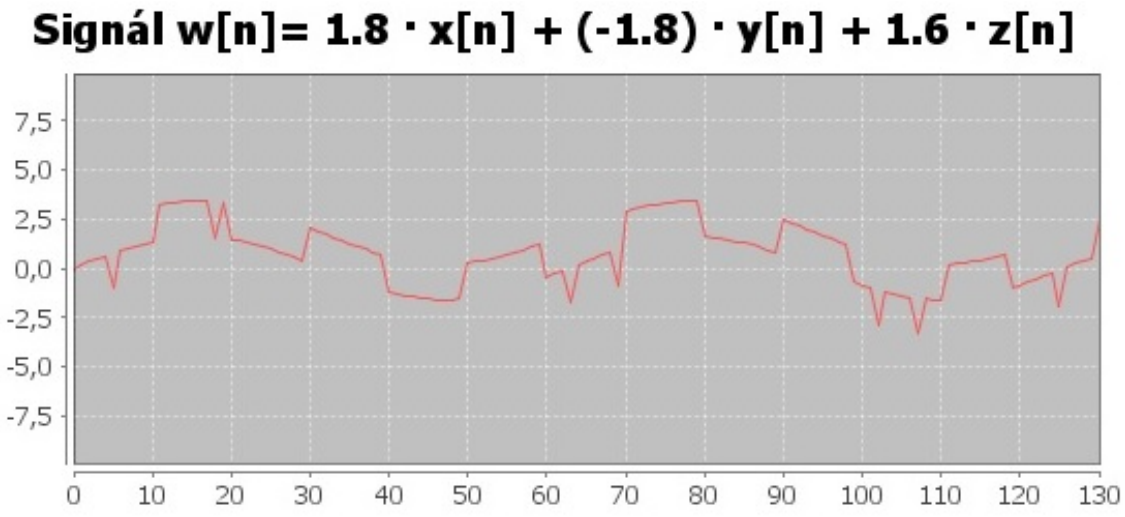


Obr. 4.8: Barycentrický trojuholník

Samotný trojuholník zobrazuje len ohraničenie, kde je možné „kliknúť“ aby sa nastavili požadované váhy signálov (signálom x, y a z odpovedajú skaláre a, b, c z kapitoly 2.4). Po kliknutí do trojuholníka sa táto zmena automaticky zobrazí aj na posuvníkoch a v trojuholníku sa znázorni kruh, ktorý označuje miesto kliku (viď obr. 4.8).

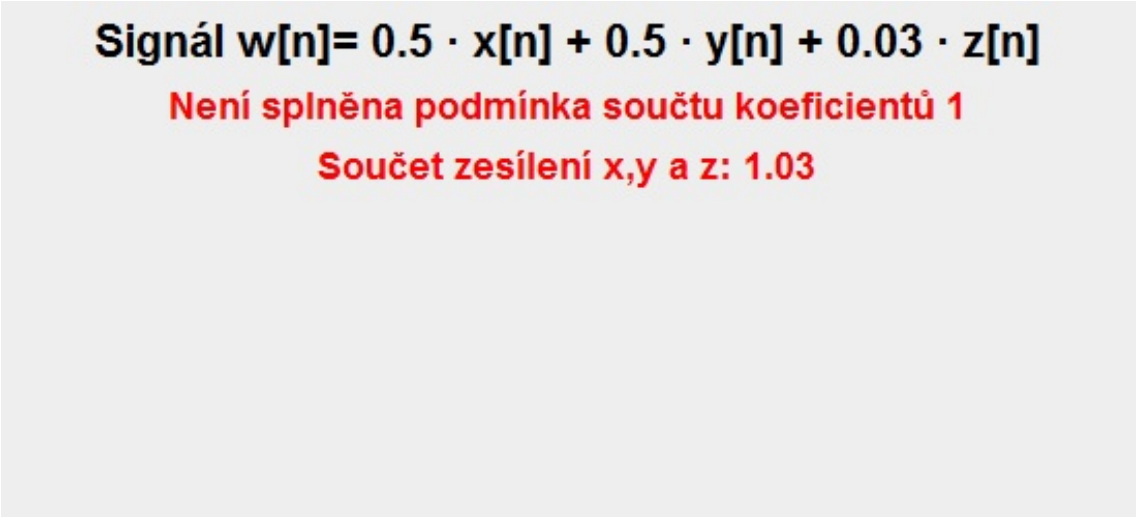
Grafy výsledných kombinácií signálov

Zvolené signály (x[n], y[n], z[n]) sú vynásobené koeficientmi určenými posuvníkmi. Potom sa tieto signály sčítajú. Výsledný graf kombinácie je možné vidieť na obr. 4.9.



Obr. 4.9: Graf výsledného signálu

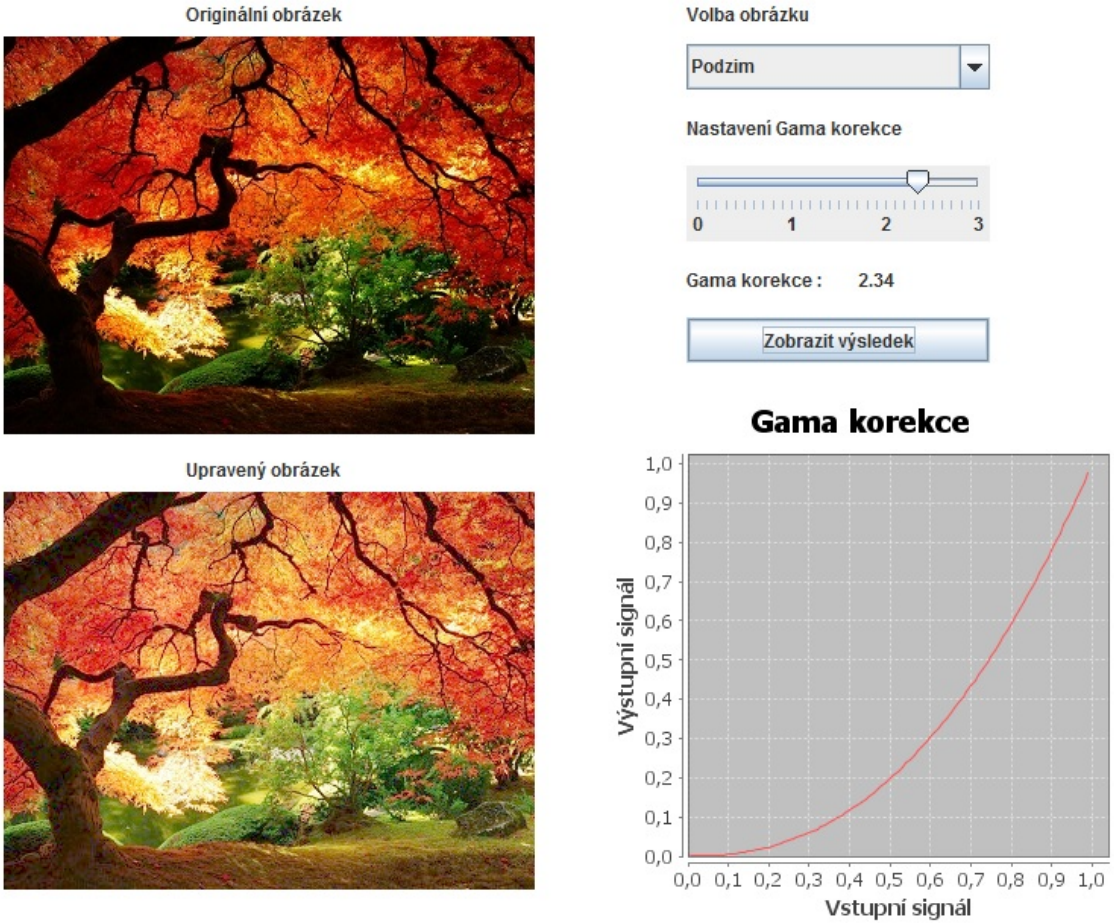
Pri vykreslovaní grafu konvexnej kombinácii musí byť najskôr splnená podmienka, že súčet všetkých koeficientov signálov je rovný 1. V opačnom prípade sa zobrazí na mieste grafu upozorňujúce hlásenie (vid obr. 4.10).



Obr. 4.10: Graf chybovej hlášky

4.2 Applet: Gama korekcia

V applete má užívateľ možnosť vybrať si z viacerých obrázkov, pri ktorých môže meniť úroveň gama korekcie. Samotný applet je možné vidieť na obr. 4.11.



Obr. 4.11: Applet gama korekcia

4.2.1 Popis appletu

Applet si môžme rozdeliť na nasledujúce časti:

1. Vstupný obrázok
2. Výstupný obrázok
3. Možnosť výberu obrázku
4. Nastavenie veľkosti gamy
5. Graf korekcie
6. Tlačidlo na zobrazenie výsledku

Možnosť výberu obrázku

Na výber máme 8 rôznych obrázkov (viď obr. 4.12).

Obr. 4.12: Výber obrázku

Po vybratí obrázku sa automaticky zobrazí nový a teda zmení predchádzajúci nastavený obraz.

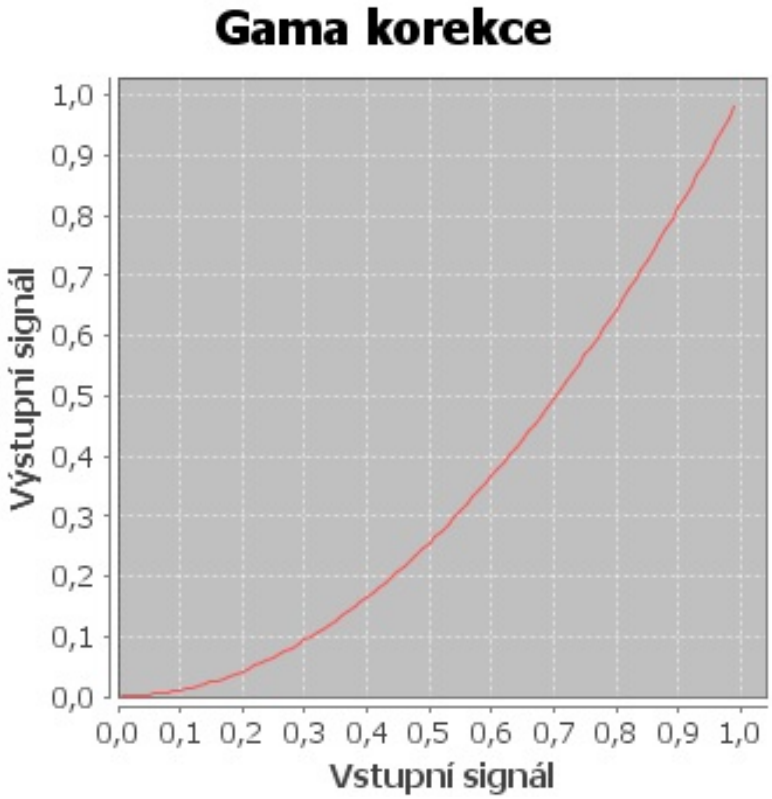
Nastavenie veľkosti gamy

Pri zapnutí appletu je prednastavená hodnota gama korekcie na hodnotu 2, 2. Veľkosť gamy korekcie nastavujeme pomocou posuvníka (viď obr. 4.13), ktorý ma rozsah od 0 do 3 pri citlivosti 0, 01.

Obr. 4.13: Výber obrázku

Graf korekcie

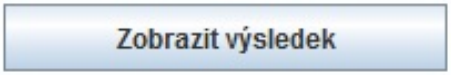
Na obr. 4.14 je možné vidieť graf gamy korekcie, ktorý zobrazuje zmenu výstupného obrazu podľa veľkosti zvolenej hodnoty gamy (nastavuje sa pomocou posuvníka).



Obr. 4.14: Graf gamy korekcie

Tlačidlo na zobrazenie výsledku

Po zvolení veľkosti gamy korekcie si pomocou tohto tlačidla zobrazíme výslednú úpravu obrázku a zároveň aj graf korekcie. Tlačidlo na zobrazenie výsledku je možné vidieť na obr. 4.15.



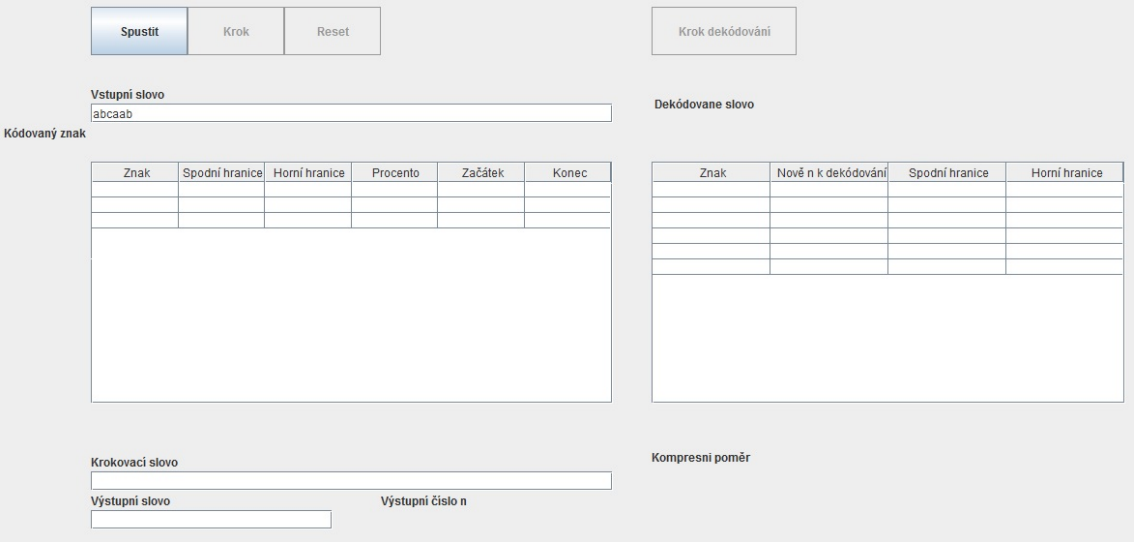
Obr. 4.15: Tlačidlo na zobrazenie výsledku

4.3 Applet: Aritmetické kódovanie a dekódovanie

V applete je možné vyskúšať si aritmetické kódovanie zadaním vstupného slova a sledovať ako sa mení výsledný interval každým krokom vypočítavania. Pri zadávaní vstupného slova, ale platia 2 podmienky a to:

1. Vstupné slovo môže tvoriť maximalne 13 rôznych znakov
2. Vstupné slovo môže mať maximalne 15 znakov, ale len za predpokladu, že platí prvá podmienka.

Po skončení kódovacieho procesu je možné správu dekódovať. Pri dekódovani je opäť možné sledovať dekódovací proces krok po kroku.



Obr. 4.16: Applet aritmetické kódovanie a dekódovanie

4.3.1 Popis appletu

Applet si môžeme rozdeliť na 4 hlavné časti:

1. Ovládacie tlačidlá
2. Vstupné pole
3. Štatistické údaje
4. Výstupné pole

Po štarte appletu je vo vstupnom poli označenom ako „Vstupní slovo“ prednastavené „abcaab“ (viď obr. 4.16). Tlačidlom „Spustit“ spúšťame celý applet. Automaticky sa vypíše štatistická tabuľka (viď obr. 4.17) obsahujúca informácie o početno-

Znak	Spodní hranice	Horní hranice	Procento	Začátek	Konec
a	0.0	0.5	50.0		
b	0.5	0.833333333...	33.33333333...		
c	0.833333333...	0.999999999...	16.66666666...		

Obr. 4.17: Štatistické údaje

sti, hodnote spodnej a hornej hranici intervalu pre každý znak. Po kliknuti na tlačidlo „Spustit“ sa toto tlačidlo deaktivuje a zruší sa možnosť upravovať vstupné slovo, ale aktivuje sa tlačidlo „Krok“ a „Reset“, tlačidlo „Krok dekódováni“ ostáva deaktivované.

Kódovaný znak abcaab

Znak	Spodní hranice	Horní hranice	Procento	Začátek	Konec
a	0.0	0.5	50.0	0.388888888...	0.395833333...
b	0.5	0.833333333...	33.33333333...	0.395833333...	0.400462962...
c	0.833333333...	0.999999999...	16.66666666...	0.400462962...	0.402777777...

Obr. 4.18: Štatistické údaje 2

Pomocou tlačidla Krok následne posúvame celý výpočet po jednom kroku. Pri každom kliknutí na tlačidlo „Krok“ sa menia štatističké údaje v tabuľke teraz už obsahujúcej aj informácie o aktuálnom rozsahu intervalu. Zároveň sa zobrazuje aktuálne kódovaný znak (na obrázku znázornený červene viď obr. 4.18).

Krokovací slovo

01100100101111011010000100101111011010000100101111010 0

Kompresni poměr

0.09375

Výstupní slovo

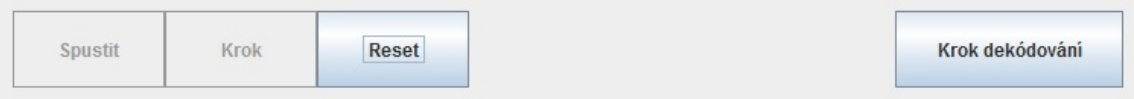
011001001

Výstupní číslo n

0.3935185185185185

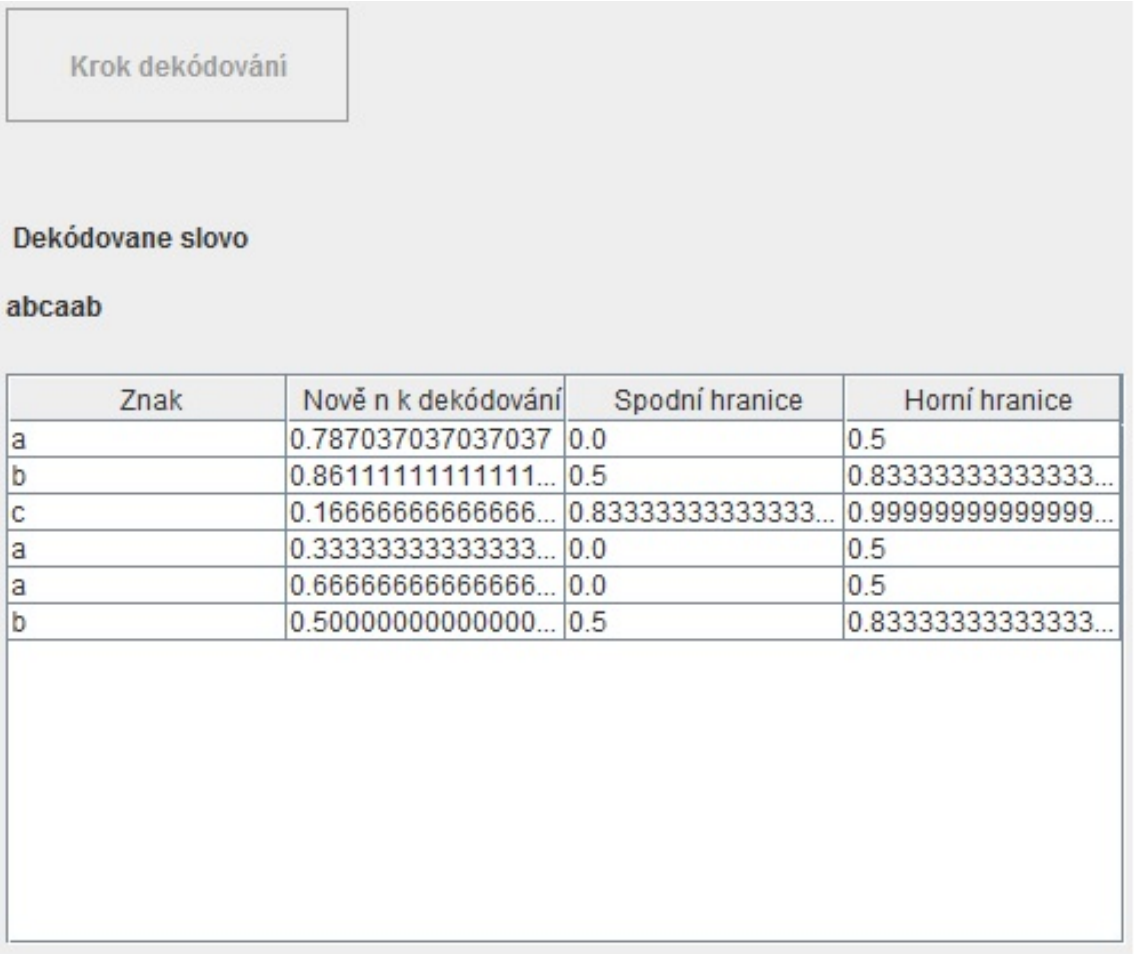
Obr. 4.19: Výstupné údaje

Ak dôjde kódovací proces na koniec, tlačidlo „Krok“ sa deaktivuje. V tomto momente sa aktivujú tlačidla „Reset“ a „Krok dekódováni“. Zároveň sa nám vypíše aj výstupne slovo, výstupné číslo n a kompresný pomer. Výsledné slovo a krokovacie slovo nemá vždy totožnú hodnotu. Výsledné slovo sa orezáva na zmenšený počet bitov tak, aby každé číslo z výsledneho intervalu mohlo byť vyjadrené rovnakou bitovou postupnosťou(viď obr. 4.19). Nakoniec môžme všetko vyresetovať pomocou tlačidla „Reset“ alebo spustiť dekódovanie(viď obr. 4.20).



Obr. 4.20: Tlačidlá

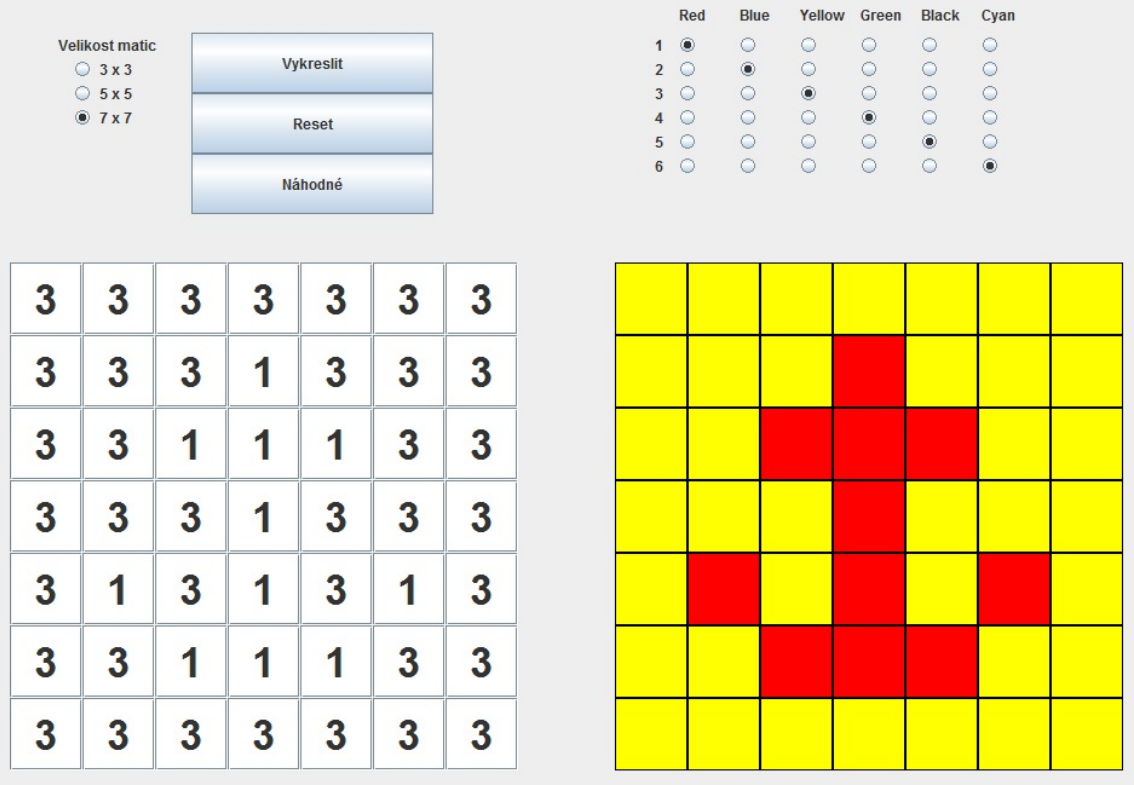
Dekódovanie prebieha vždy po jednom kroku a to stlačením tlačidla „Krok dekódováni“.Pri každom stlačení tlačidla sa vypíše aktuálny dekódovaný znak, ktorý sa automatický vypíše aj v tabuľke spolu so spodnou a hornou hranicou intervalu, do ktorého daný znak spadá. Následne sa vypočíta nové n. Pokial dôjde dekódovací proces na koniec, tlačidlo dekódovací krok sa deaktivuje(viď obr. 4.21) a ako jediné aktívne tlačidlo ostane „Reset“. Stlačením „Reset“ sa celý applet vráti do pôvodného stavu.



Obr. 4.21: Dekódovanie

4.4 Applet: Indexácia farby

V tomto applete si môžeme vyskúšať indexáciu farieb, kde je možné priradiť rôznym číslam rôzne farby a následne to vykresliť v matici. Samotný applet je možné vidieť na obr. 4.22.



Obr. 4.22: Applet Indexácie farieb

4.4.1 Popis appletu

Applet môžeme rozdeliť na nasledujúce časti:

1. výber veľkosti matice
2. indexovanie farieb
3. ovládacie tlačidlá
4. vykresľovacie matice

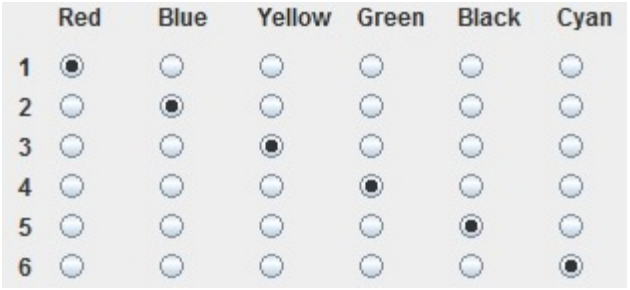
Výber veľkosti matice

Na obr. 4.23je možné vidieť, že na výber matice máme tri možnosti. Pôvodne je nastavená matica 7x7.

Obr. 4.23: Výber veľkosti matíc

Indexovanie farieb

Užívateľ má možnosť prideliť 6 rôznym číslam 6 rôznych farieb. Každému číslu môže byť pridelená maximálne 1 farba, ale 1 farba môže byť pridelená viacerým číslam. Pri štarte appletu je nastavená indexácia farieb pomocou diagonály, ako je možné vidieť na obr. 4.24



Obr. 4.24: Výber farieb

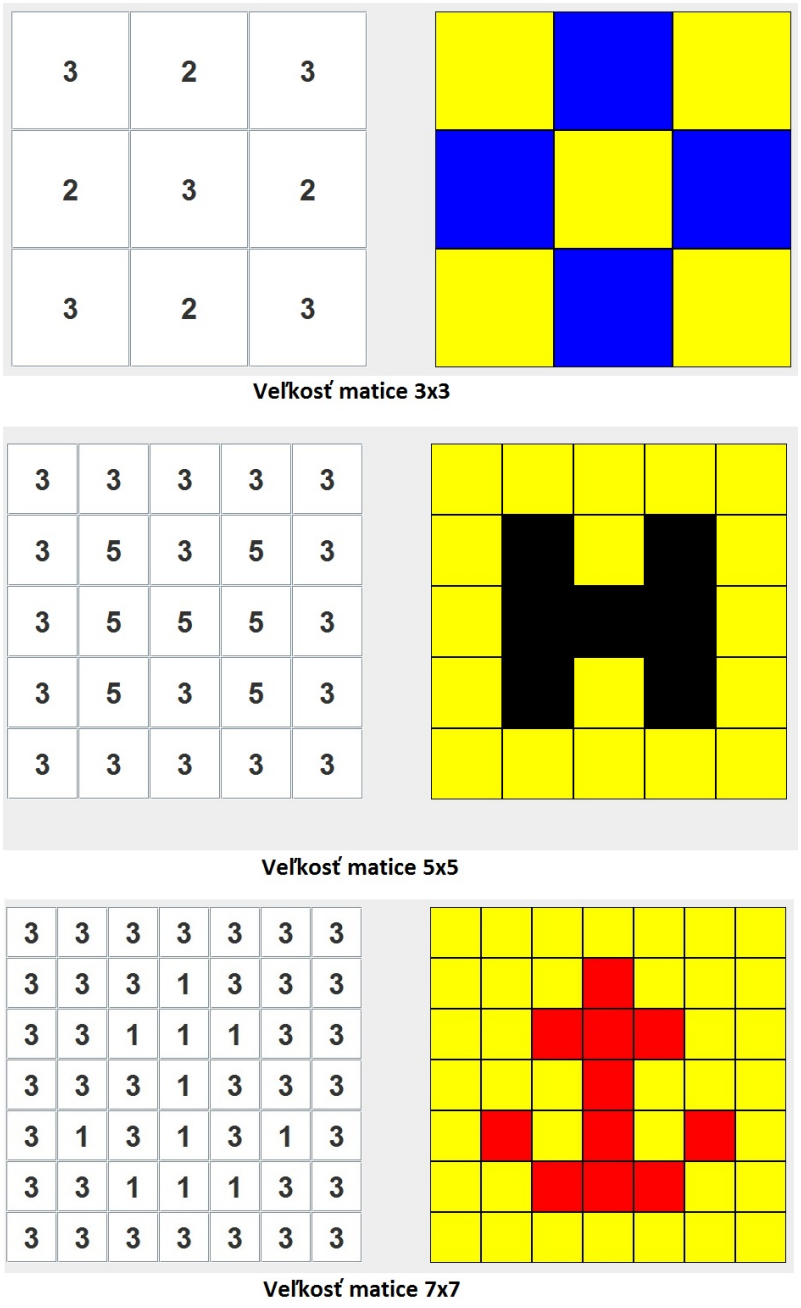
Ovládacie tlačidlá

Pri štarte appletu si užívateľ môže zvoliť veľkosť matice, prideliť číslam určité farby a následne ich môže zapísať do editovateľnej matice. Ak užívateľ chce vykresliť maticu je nutné stlačiť tlačidlo „Vykreslit“. Ak sa mu obrazec nepozdáva, môže ho celý vymazať pomocou tlačidla „Reset“. Ak užívateľ nemá jasnú predstavu vyplnenia matice, má možnosť kliknúť na tlačidlo „Náhodné“ (viď obr. 4.25) a matica sa vyplní náhodnými číslami (farby musí nastaviť sám).

Obr. 4.25: Ovládacie tlačidla

Vykresľovacie matice

Pre každú veľkosť matice je nastavené prednastavené vykreslenie (viď obr. 4.23). Matica obsahujúca čísla je editovateľná. Užívateľ má možnosť sám zmeniť čísla podľa jeho vlastnej predstavy.



Obr. 4.26: Defaultne zobrazené matice

5 ZÁVER

Výsledkom mojej diplomovej práce sú 4 funkčné webové applety napísané v programe NetBeans, ktoré v budúcnosti budú slúžiť ako podpora vo výuke. Vytvorené sú nasledujúce applety:

Applet znázorňujúci kombinácie signálov

V applete je možné simulovať lineárnu a konvexnú kombináciu signálov. Popis jednotlivých kombinácií je možné vidieť v kapitole 2. Užívateľ si môže vybrať zo 6 typov signálov, pri ktorých je možné meniť veľkosť zvolených signálov pomocou posuvníkov. Pri lineárnej kombinácii je daný rozsah od -3 do +3 a pri konvexnej kombinácii je rozsah od 0 do 1. Bližší popis fungovania appletu sa nachádza v kapitole 4.1.

Applet na úpravu obrazu pomocou gama korekcie

V applete si môže užívateľ zvoliť 1 z 8 rôznych obrázkov, na ktorých je možné vyskúšať zmenu zobrazenia obrázka pri rôznych úrovniach gamy korekcie. Veľkosť gamy korekcie je možné regulovať na posuvníku, ktorý má rozsah od 0 do 3. Užívateľ má možnosť porovnávať upravený obrázok s originálnym obrázkom. V grafe je možné vidieť krivku gamy korekcie. Bližší popis fungovania appletu sa nachádza v kapitole 4.2.

Applet znázorňujúci aritmetické kódovanie a dekódovanie

Ako už názov napovedá v tomto applete je možné si vyskúšať aritmetické kódovanie a dekódovanie. Po zadaní vstupného slova (možné použiť celú abecedu aj číselné znaky, ale maximálne 15 znakov a tento reťazec môže obsahovať maximálne 13 rôznych znakov) môže užívateľ sledovať po 1. kroku ako prebieha samotné kódovanie či dekódovanie. V tabuľkách sa zobrazujú intervaly pre každý použitý znak, v ktorom sa daný znak momentálne nachádza. Výstupné slovo je zobrazené v spodnej časti appletu v binárnej podobe spolu s výstupným číslom n a kompresným pomerom. Bližší popis fungovania appletu sa nachádza v kapitole 4.3.

Applet s indexáciou farieb

Pri tomto applete má užívateľ možnosť zvoliť si jednu z 3 veľkostí vykresľovanej matice. V editovateľnej matici je môžné zadávať čísla od 1 do 6. Ku každému číslu je môžné prideliť iba 1 z 6 rôznych farieb, ale zároveň platí, že 1 farbu môže prideliť ku všetkým číslam. Bližší popis fungovania appletu sa nachádza v kapitole 4.4.

Všetký applety sú umiestnené na stránke vedúceho práce:
`http://www.utko.feec.vutbr.cz/~rajmic/applets/`

LITERATÚRA

[1] *Aritmetické kódování* [online]. [cit. 2014-05-26]. Dostupné z: <http://martin.lipinsky.cz/skola/pt/HTML/54/default.htm>

[2] BOBOT, V a JAKUBEKOVÁ,M. *Interaktívne vyučovanie v školských vzdelávacích programoch* [online]. [cit. 2014-05-26]. Dostupné z: <http://www.mpc-edu.sk/library/files/publik_ciaivvsvp_finalversion_na_web-29_10_2012.pdf>

[3] BUSA, J. *Mini minimalizácia* [online]. 2011 [cit. 31. 12. 2013]. Dostupné z URL: <http://web.tuke.sk/fei-km/old/OM/OMa4.pdf>

[4] CATTIN, P.: *Image Restoration: Introduction to Signal and Image Processing* [online]. 2008 [cit. 31. 12. 2013]. Dostupné z URL: <http://miac.unibas.ch/SIP/06-Restoration.html>

[5] HORDĚJČUK, V. *Aritmetické kódování* [online]. [cit. 2014-05-21]. Dostupné z: <http://old.voho.cz/wiki/informatika/kodovani/aritmeticke/>

[6] *JFreeChart: Welcome To JFreeChart!* [online]. 2005 [cit. 31. 12. 2013]. Dostupné z URL: <http://www.jfree.org/jfreechart/>

[7] JULES, C. *Accurate point in triangle test* [online].25. január 2014 [cit. 2014-05-21] . Dostupné z: <http://totologic.blogspot.fr/2014/01/accurate-point-in-triangle-test.html>

[8] KOTALA, Z. a P. TOMAN. *Applet* [online]. 04. február 2001 [cit. 31. 12. 2013]. Dostupné z URL: <http://v1.dione.zcu.cz/java/sbornik/17.html>

[9] MCGUFFIN, M. *Exercise 1: Drawing Lines* [online]. 2012 [cit. 2014-08-23]. Dostupné z: <http://www.javakode.com/applets/01-drawingLines/>

[10] MORKES, D. *Java Applet krok za krokem* [online]. 27. september 2002 [cit. 31. 12. 2013]. Dostupné z URL: <http://interval.cz/clanky/java-applet-krok-za-krokem/>

[11] *NetBeans* [online]. 2013 [cit. 31. 12. 2013]. Dostupné z URL: <https://netbeans.org/index_cs.html>

[12] ONDREJKA, Petr. *Aritmetické kódování: Komprimace dat a kryptologie* [online]. 20. december 2012 [cit. 2014-08-23]. Dostupné z: <https://akela.mendelu.cz/∼xondrejk/KAS/aritmeticke_kodovani.pptx>

[13] POYNTON, Charles. *The rehabilitation of gamma* [online]. [cit. 2014-05-21]. Dostupné z: <http://www.poynton.com/PDFs/Rehabilitation_of_gamma.pdf>

[14] *Programovací jazyk Java*^{TM} [online]. 2013 [cit. 31. 12. 2013]. Dostupné z URL: <http://v1.dione.zcu.cz/java/uvod.html>

[15] STANEK, M. *Aritmeticke kodovanie a bwt: Kódovanie a kryptológia.* 2002.

[16] TULEJA, S. GYMNÁZIUM ARM. GEN. L. SVOBODU, Humenné. *Využitie Java appletov vo vyučovaní fyziky* [online]. 25. december 2010 [cit. 2014-08-23]. Dostupné z: <http://www.stuleja.org/vscience/seminar/pedagogikaAppletov.pdf>

[17] VODIČKOVÁ, V.: *Lineárna kombinácia vektorov* [online]. 4. apríl 2013 [cit. 31. 12. 2013]. Dostupné z URL: <http://www.sportgymke.sk/mvd/AnalytickaGeometria/LinearnaKombinacia-Vektorov.pdf >

[18] WEISSTEIN, E. *Barycentric Coordinates* [online]. 16. máj 2014 [cit. 2014-05-21]. Dostupné z: <http://mathworld.wolfram.com/BarycentricCoordinates.html>

A PRÍLOHY

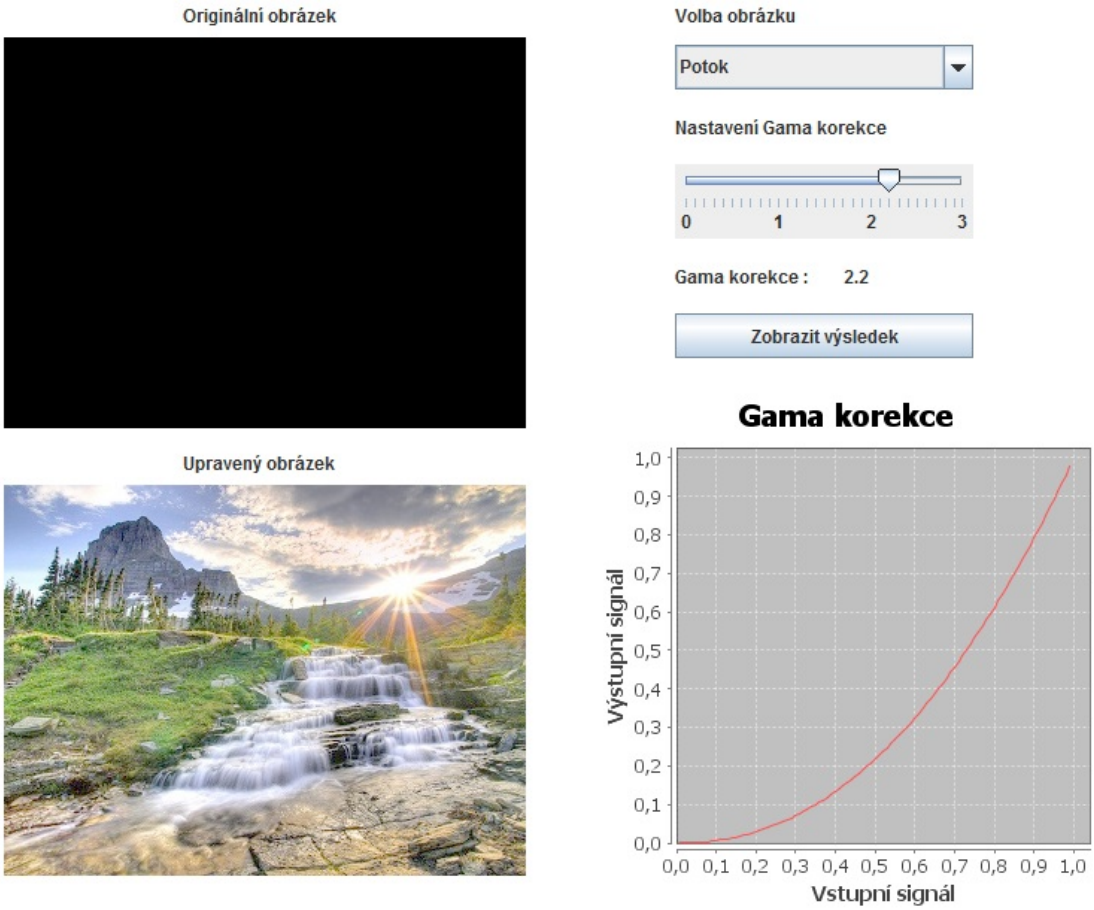
A.1 Možné problémy

Pri vytváraní appletu znázorňujúceho gama korekciu som narazil na niekoľko chýb.

1. Pod operačným systémom
   - Windows 8
   - Windows 8.1
   - Windows Vista
2. Nastavenie Java security

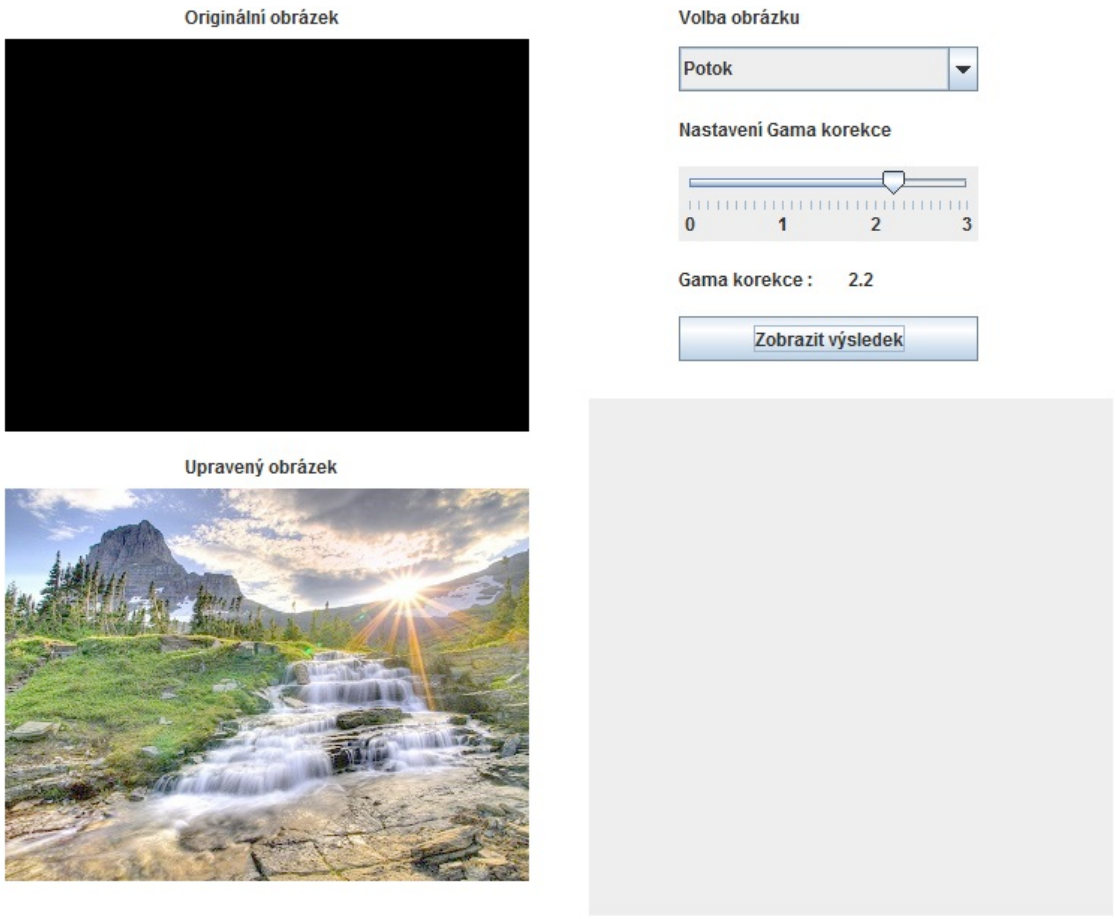
A.1.1 Pod operačným systémom

Je chyba vo vykresľovaní appletu po spustení, resp. pri zmene z pôvodne nastaveného obrázka na vybraný obrázok (viď obr. A.1).



Obr. A.1: Chybné vykreslenie appletu 1

Zároveň pri kliknutí na tlačídlo „Zobrazit výsledek“, opätovne nevykreslí všetky obrázky korektne (viď obr. A.2).



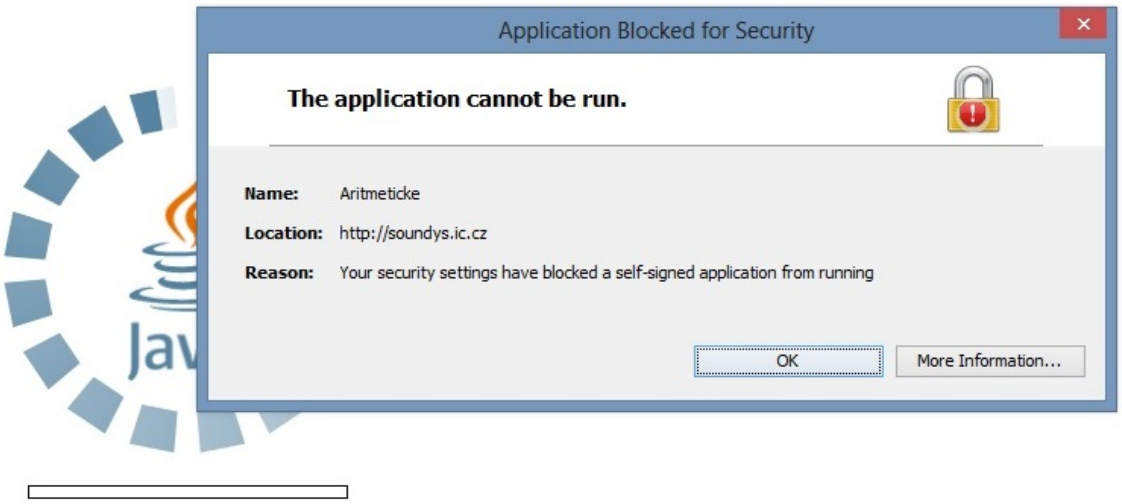
Obr. A.2: Chybné vykreslenie appletu 2

A.1.2 Nastavenie Java security

Pokiaľ je na PC nainštalovaná najnovšia verzia JAVY (dnešným dňom je vydaná verzia 7u51), nastáva problém s bezpečnostnými pravidlami JAVY, ktoré sú implementované s najnovšou verziou.

Po spustení akéhokoľvek appletu Vám vyskočia 2 po sebe vyskakovacie okná znemožňujúce spustenie appletu (viď obr. A.3 a obr. A.4).

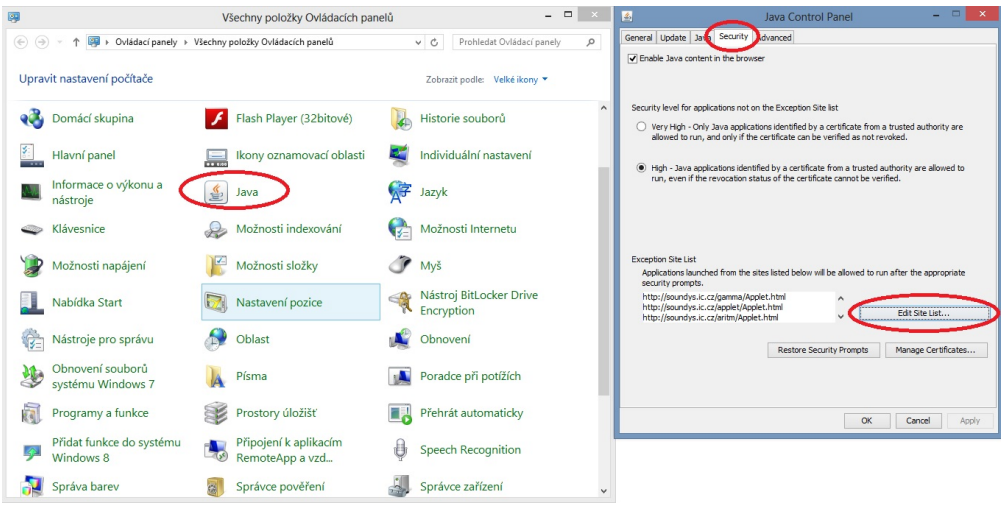
Obr. A.3: Blokovanie appletu 1



Obr. A.4: Blokovanie appletu 2

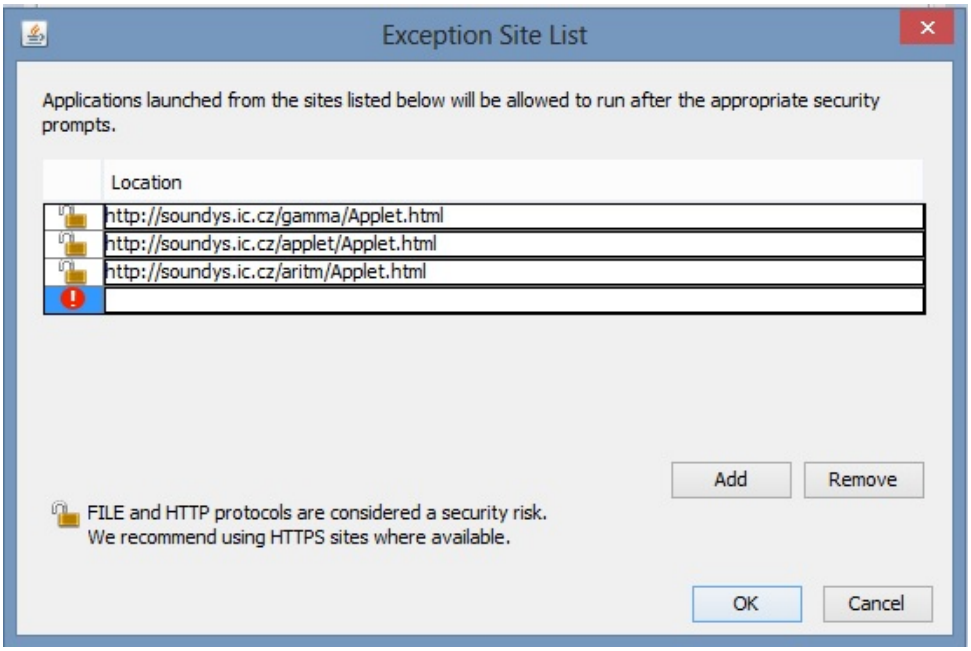
Na zobrazenie appletu je nutné spraviť nasledujúce kroky.

1. Otvoriť ovládacie panely → Java → Security → Edit Side List (viď obr. A.5).



Obr. A.5: Nastavanie JAVA security 1

2. Po kliknutí na Edit Side List sa objaví nižšie uvedené okno, v ktorom je potrebné kliknúť na tlačidlo „Add“ a do voľného riadku zadať URL adresu daného Java Appletu a potvrdiť tlačidlom OK (viď obr. A.6).



Obr. A.6: Nastavanie JAVA security 2

3. Po potvrdení vyskočí oznamovacie okno, kde posledný krát potvrdíte dôveryhodnosť daného appletu (viď obr. A.7).

Obr. A.7: Nastavanie JAVA security 3

B PRÍLOHY

B.1 obsah CD

Priložené CD obsahuje zdrojové kódy ku všetkým 4 appletom a obrázky použité v jednotlivých appletoch.

Cesta ku zdrojovým kódom a knižniciam pre jednotlivé applety je nasledujúca:

1. applet na simuláciu lineárnej a konvexnej kombinácie signálov
   - ./applety/Kombinacie/dist/Kombinacie.jar
   - ./applety/Kombinacie/dist/lib
   - ./applety/Kombinacie/Applet.Html
2. applet na úpravu obrazu pomocou gama korekcie
   - ./applety/GamaKorekcia/dist/GamaKorekcia.jar
   - ./applety/GamaKorekcia/dist/Lib
   - ./applety/GamaKorekcia/Applet.Html
3. applet znázorňujúci aritmetické kódovanie
   - ./applety/Aritmeticke/dist/Aritmeticke.jar
   - ./applety/Aritmeticke/Applet.Html
4. applet s indexáciou farieb
   - ./applety/IndexovaneFarvy/dist/IndexovaneFarvy.jar
   - ./applety/IndexovaneFarvy/Applet.Html

Pri nahrávaní appletu gama korekcia a appletu na simuláciu lineárnej a konvexnej kombinácie signálov na FTP server je nutné skopírovať do koreňového adresára zložku „lib“ obsahujúcu dodatočné knižnice. U všetkých appletov je nutné skopírovať do koreňového adresára taktiež html súbor ./daný_applet/Applet.html.

C PRÍLOHY

C.1 Ukážky kódu

V tejto časti sa nachádzajú ukážky použitého kódu.

C.1.1 Vytvorenie data-setu pre graf

```
1.      XYSeriesCollection dataset = new XYSeriesCollection();
2.      dataset.addSeries(Values);
3.       JFreeChart chart = ChartFactory.createXYLineChart(pojmenovani , // Title
4.          , // x-axis Label
5.          , // y-axis Label
6.          dataset, // Dataset
7.          PlotOrientation.VERTICAL, // Plot Orientation
8.          false, // Show Legend
9.          false, // Use tooltips
10.           false // Configure chart to generate URLs?
11.           );
12.       chart.setAntiAlias(true);
13.       XYPlot xyPlot = chart.getXYPlot();
14.       org.jfree.chart.axis.ValueAxis domainAxis = xyPlot.getDomainAxis();
15.       org.jfree.chart.axis.ValueAxis rangeAxis = xyPlot.getRangeAxis();
16.       domainAxis.setRange(0.0, 130);
17.       if(view.isVisible())
18.          rangeAxis.setRange(-1.3, 1.3);
19.       else
20.          rangeAxis.setRange(-3.3, 3.3);
21.       ChartPanel CP = new ChartPanel(chart);
22.       panel.add(CP,BorderLayout.CENTER);
23.       panel.validate();
24.       if(Graf == 1)
25.          System.arraycopy(hodnoty, 0, hodnotyOne, 0, 131);
26.       if(Graf == 2)
27.          System.arraycopy(hodnoty, 0, hodnotyTwo, 0, 131);
28.       if(Graf == 3)
29.          System.arraycopy(hodnoty, 0, hodnotyThree, 0, 131);
30.       return panel;
```

Na začiatku si naplníme „dataset“ hodnotami, ktoré chceme použiť pri vykreslení grafu. Riadky 3-12 slúžia na vytvorenie novej datovej štruktúry grafu, kde nastavuje rôzne parametre grafu ako je pomenovanie os, zobrazovanie legendy a podobne. Následujúcimi riadkami 13-15 vytvoríme nové dátové štruktúry xyPlot slúžiace na úpravu os grafu. Samotný rozsah os nastavujeme pomoc riadkov 16-20. V nasledúcich riadkoch (21-23) si vytvorím chartPanel, ktorý vložím do navratového panelu a potvrdime vykreslenie panelu. Posledná časť kódu(riadky 24-29) slúži na uloženie hodnôt grafov pre neskoršie porovnanie využívané pri konvexnej konvolúcii. Riadok 30 navracia hodnotu panelu.

C.1.2 Dekódovacia metóda

```
1.    public String Decode(double n, int delkaSlova)
2.    {
3.       String s =" ";
4.       double low = 0;
5.       double hight = 1;
6.       int citac = 0;
7.       double range = 1;
8.       while (citac < delkaSlova)
9.       {
10.           for(int i = 0; i < Prvky.size();i++)
11.           {
12.              if(n >= Prvky.get(i).GetLow() && n < Prvky.get(i).GetHight() )
13.               {
14.                  low = Prvky.get(i).GetLow();
15.                 hight = Prvky.get(i).GetHight();
16.                 range = hight-low;
17.                 s = s + Prvky.get(i).GetZnak();
18.                 n = (n - low)/range;
19.                 break;
20.               }
21.           }
22.           citac++;
23.       }
24.       return s;
25.    }
```

Vstupom pre dekódovanie je štartovacie číslo a dĺžka slova. V 4. až 7. riadku nastavujem vstupné premenné. Následne prechádzam prvky v zozname(10) a pokiaľ platí pre niektorý prvok podmienka(12), že „n“ leží v jeho intervale, prepočítajú sa hodnoty dolnej a hornej hranice, nastaví sa východzí interval „s“ a vypočíta sa nové „n“ (14-18). Nakoniec dochádza k prerušeniu cyklu (19), zvýši sa počet dekodovaných znakov o 1 a navracia sa hodnota „s“.

C.1.3 Kódovací algoritmus

```
1.      double intervalStart = 0;
2.      double intervalKonec = 1;
3.      for(int i = 0; i < pole.length;i++)
4.      {
5.          double intervalCelek = intervalKonec - intervalStart;
6.          double downCut = 0;
7.          for(int j = 0; j < Prvky.size();j++)
8.          {
9.              if (Prvky.get(j).GetZnak()==pole[i])
10.             {
11.                 for(int k = 0; k < j; k++)
12.                 {
13.                     double oriznuti = Prvky.get(k).GetPomer();
14.                     downCut =downCut + oriznuti;
15.                 }
16.                 double oriznuti = Prvky.get(j).GetPomer();
17.                 double spodni = downCut * intervalCelek;
18.                 double interval = oriznuti * intervalCelek;
19.                 intervalStart = intervalStart + spodni;
20.                 intervalKonec = intervalStart + interval;
21.             }
22.         }
23.          nHodnotaStart=(intervalKonec-intervalStart)/2+intervalStart;
24.     }
```

Na začiatku si nastavím štartovacie hodnoty intervalu pre všetky znaky, ktoré sú v reťazci(1-3). Vypočítam opätovne interval. Nastavím spodné oreznutie intervalu. V 7 riadku začínam hľadať prvok, ktorý odpovedá aktuálne kódovanému znaku. Konretný prvok nachádzam vďaka riadku 9 a hneď spočítavam o koľko sa zmení spodná hranica intervalu(11-15). Riadok 16 nám určí o koľko sa interval oreže kvôli

súčasnému znaku, v riadku 17 kvôli predchádzajúcim znakom a v riadku 18 sa určí akú časť z aktuálneho rozsahu predstavuje aktuálne kódovaný znak. Opätovne prenastavím hodnoty hraníc intervalu(19-20). V riadku 23 určujem výsledné číslo „n“, ktoré budem prevádzať z dekadického tvaru na binárny tvar.

C.1.4 Výsledné slovo

```
1.      double cislo =((Math.log(1/(intervalKonec-intervalStart)))/Math.log(2))+1;
2.      double zbytek = intervalStart+(intervalKonec-intervalStart)/2;
3.      String zprava = " ";
4.       while (zbytek != 1)
5.      {
6.         zbytek = zbytek*2;
7.         if(zbytek > 1)
8.            {
9.               zprava = zprava + "1";
10.               zbytek = zbytek-1;
11.            }
12.         else
13.            zprava = zprava + "0";
14.      }
15.      if(cislo != Double.POSITIVE_INFINITY && cislo !=Double.NEGATIVE_INFINITY
&& (((int) cislo) < zprava.length()))
16.         {
17.            String zkraceny = zprava.substring(0, (int) cislo);
18.            zprava = zkraceny;
19.         }
20.      vysledneSlovo.setText(zprava);
```

V prvom riadku si vypočítam na koľko ma byť orezaný výsledny reťazec. Nastavím premenu „zbytek“ na polovicu intervalu medzi hornou a spodnou hranicou intervalu(aby nedochádzalo na hranách intervalu k pretekaniu do inej premenej kvôli zaokrúhľovaní pri dekódovacom procese). Následne v riadkoch 4-14 prebieha samotný prevod z dekadického tvaru na binárny. V 15. riadku sa nachádza podmienka, ktorá určuje, že pokiaľ nie je počet znakov, na ktoré sa má orezať nekonečný a zároveň je kratší ako kódovaná správa, dôjde k orezaniu správy na určený počet znakov. Nakoniec si pomocou 20. riadku vypíšem výsledné slovo.